НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«До захисту допущено»
Завідувач кафедри
Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система дистанційного навчання іноземної мови»

Виконав:
Студент IV курсу, групи ІО-62
Горбов Олексій Юрійович _____

Керівник:
ст.викл.
Виноградов Юрій Миколайович _____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., проф.
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій СТІРЕНКО

«___»_____2020 р.

ЗАВДАННЯ

на дипломний проект студенту

Горбову Олексію Юрійовичу

1. Тема проекту «Система дистанційного навчання іноземної мови», керівник проекту Виноградов Юрій Миколайович, старший викладач, затверджені наказом по університету від «07» травня 2020 р. № 1081-с
2. Термін подання студентом проекту *26 травня 2020р.*
3. Вихідні дані до проекту *див. технічне завдання*
4. Зміст пояснювальної записки *Аналіз і характеристика об'єкта проектування, обґрунтування оптимального варіанта реалізації мети цієї роботи, розробка додатку: вибір технологій та їх обґрунтування, основні рішення з реалізації додатку. Висновки*
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) *принципова схема, функціональна схема, структурна схема.*
6. Консультанти розділів проекту.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П., професор, д.т.н.		

7. Дата видачі завдання 01.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Затвердження теми роботи	01.09.2019	
2.	Вивчення та аналіз завдання	15.12.2019-15.03.2020	
3.	Розробка архітектури додатку	15.03.2020-25.03.2020	
4.	Написання програмної частини	25.03.2020-05.04.2020	
5.	Тестування та виправлення помилок	05.04.2020-15.04.2020	
6.	Оформлення пояснювальної записки	15.04.2020-20.05.2020	
7.	Захист програмного продукту	25.04.2020	
8.	Передзахист	26.05.2020	
9.	Захист	18.06.2020	

Студент Олексій ГОРБОВ

Керівник Юрій ВИНОГРАДОВ

Анотація

Робота присвячена розробці системи дистанційного навчання іноземної мови використовуючі сучасні веб-технології. Задля того щоб покращити систему було вирішено застосувати деякі алгоритми обробки природної мови, а також зовнішні програмні інтерфейси для перекладу та пошуку зображень.

Серверна частина була розроблена за допомогою Django, а для написання клієнтської частини було обрано фреймворк VueJS.

Аннотация

Работа посвящена разработке системы дистанционного изучения иностранного языка используя современные веб-технологии. Для того чтобы улучшить систему было решено использовать некоторые алгоритмы обработки естественного языка, а также внешние программные интерфейсы для перевода и поиска изображений.

Серверная часть была разработана с помощью Django, а для написаня клиентской части было выбрано фреймворк VueJS.

Abstract

The work is devoted to the development of a system of remote learning a foreign language using modern web technologies. In order to improve the system, it was decided to use some natural language processing algorithms and external services for translating text and searching images.

The server part was developed using Django, and the VueJS framework was chosen for writing the client part.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	6505.02.000 ТЗ	Технічне завдання	3	
3	А4	6505.03.000 ПЗ	Пояснювальна записка	55	
4	А4	6505.04.000 Д1	Структурна схема	1	
5	А4	6505.05.000 Д2	Функціональна схема	1	
6	А4	6505.06.000 Д3	Принципова схема	1	

Зм.	Арк.	№ докум.	Підпис	Дата	ДП.6505.01.000 ВП		
Розробив		Горбов О.Ю.			*Система дистанційного навчання іноземної мови Відомість проекту*	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Виноградов Ю.М.					
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-62	
Н. Контр.		Сімоненко В. П.					
Затвердив							

Технічне завдання
до дипломного проекту

На тему «Система дистанційного навчання іноземної мови»

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Система дистанційного навчання іноземної мови» Область застосування: програму можуть використовуватися студенти які хочуть покращити свої знання в іноземних мовах.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення системи дистанційного навчання іноземної мови

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по платформах дистанційного навчання, публікації в мережі Інтернет по даній темі.

5. ТЕХНІЧНІ ВИМОГИ

Додаток, що розробляється повинен бути зручним у використанні. Повинен мати зрозумілий інтерфейс, із швидкою навігацією по системі.. Потрібно, щоб була висока швидкість відгуку на запити користувачів.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення необхідної літератури	19.02.2020
Складання і узгодження технічного завдання	06.03.2020
Написання вступної частини та огляд рішень	19.03.2020
Розробка архітектури додатку	03.04.2020
Написання програмної частини	10.04.2020
Тестування та виправлення помилок	01.05.2020
Оформлення документації дипломного проекту	15.05.2020
Попередній захист та проходження нормативного контролю	29.05.2020
Захист дипломного проекту	18.06.2020

Пояснювальна записка
до дипломного проекту

на тему: Система дистанційного навчання іноземної мови

Київ - 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СДН – Система дистанційного навчання.

API – Application programming Interface.

AWS – Amazon Web Services.

CRUD – Create, Read, Update, Delete.

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol.

IP – Internet Protocol.

JSON – JavaScript Object Notation.

JWT – JSON Web Token.

SOLID – Single responsibility principle, Liskov substitution, interface segregation, dependency inversion.

TCP – Transfer Control Protocol

ВСТУП

Через глобалізацію світу зростає необхідність вивчення іноземних мов, тому також зростає і необхідність у якісних системах які дозволяють навчатися швидно та ефективно. Необхідною складовою таких систем має бути словник та вбудований або інтегрований перекладач. Було б також бажано, щоб існувало багато інтеграцій із зовнішніми сервісами, які б могли полегшити навчання, але існуючі системи майже не мають інтеграцій тому мають певні недоліки.

Мета і задачі

Метою роботи є розробка системи дистанційного навчання іноземної мови, яка обьеднує в собі усі переваги вже існуючих систем обчислювальних системах, яка також має корисні інтеграції із зовнішніми сервісами.

Для досягнення поставленої мети були поставлені наступні основні задачі:

- Створити систему у вигляді веб-додатку, який виконує базові функції системи дистанційного навчання
- Додати інтеграції із зовнішніми сервісами, для поліпшення навчання
- Протестувати працездатність усієї системи

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ СИСТЕМ ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ ІНОЗЕМНОЇ МОВИ

Системи дистанційного навчання(СДН) іноземних мов – це системи, які прискорюють і/або покращують якість вивчання іноземних мов. На сьогоднішній день, налічується багато різних таких систем, кожна з яка має як свої переваги, так і недоліки [1].

Здебільшого СДН іноземних мов мають такі складові:

- Модуль реєстрації та авторизації користувача.
- Модуль налаштувань для вибору мови, якою користувач хоче оволодіти.
- Свій вбудований перекладач або інтеграція із зовнішнім перекладачем.
- Словник слів з можливістю додавання, редагування та видалення перекладів.
- Алгоритм, завдяки якому система визначає чи вивчив користувач іноземне слово чи ні.
- Інші модулі, які допомагають користувачу якісніше та швидше оволодіти необхідною мовою.

1.1. Різновиди СДН іноземних мов

За режимом роботи СДН іноземних мов поділяються на:

- *offline*

  Це системи які працюють повністю автономно, та не потребують підключення до мережі Інтернет. Здебільшого ці системи вже не є актуальними через багаті можливості та розповсюдженість Інтернету. Приклади таких систем: SuperMemo [2].

- *online*

  В даному типі, система працює лише при підключенні до меджі Інтернет. Приклади таких систем: Moodle.

- *Hybrid (offline + online)*

  Системи, що мають гібридний режим роботи найбільш розповсюджені, вони працюють і при підключенні до Інтернету, і якщо його немає. Приклади таких систем: Anki, Memrise, та багато інших.

Рис. 1.1. Приклад роботи програми Anki

За видами алгоритмів для розподілених повторень СДН іноземних мов поділяються на[3]:

- Leitner system: довільна кількість етапів або лише 5 етапів. Метод полягає в повторенні іноземних слів через різні проміжки часу в залежності від результату відтворення слова по пам'яті.
- На основі нейронних мереж. Нейромережі набирають обертів майже у всіх сферах, тож в СДН іноземних мов вони також можуть використовуватись для алгоритмів розподілених повторень.
- Сімейство алгоритмів SM (SuperMemo)

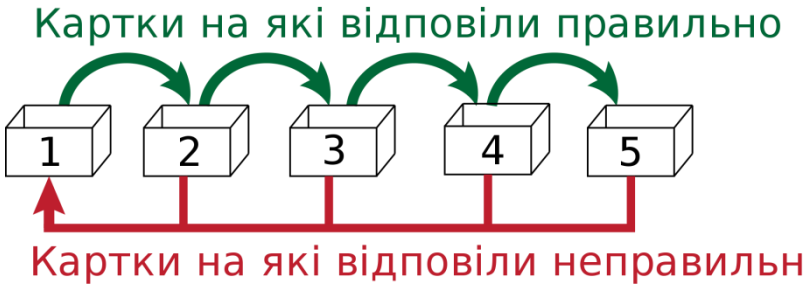


Рис. 1.2. Принцип роботи 5ти етапної системи Лейтнера[4]

1.2. Особливості створення СДН іноземних мов

Створення СДН у цілому є складним комплексним завданням, що вимагає вирішення багатьох питань щодо коректної роботи системи. Можна, з певною часткою умовності, виділити кілька проблемних областей. Серед них збереження та керування великим обсягом даних, а також, якщо СДН має велику кількість інтеграцій – підтримка цих інтеграцій. Через великий обсяг даних та інтеграцій система також потребує проектування якісної інфраструктури, яка саме і забезпечить безперервну роботу усієї системи.

1.3. Огляд та порівняння існуючих СДН іноземних мов

На сьогоднішній день налічується вже не один десяток СДН іноземних мов, зараз ми оглядово подивимося на деякі з них.

- ***Moodle***

  На сьогоднішній день Moodle безсумнівно одна з найпопулярніших СДО з відкритим вихідним кодом. Moodle пропонує користувачеві різні панелі інструментів, можливість відстежувати прогрес студентів і підтримку мультимедіа. Система дає можливість створювати курси, адаптовані під мобільні телефони, і досить доброзичливо ставиться до інтеграції доповнень від сторонніх розробників. Крім того, сервіс пропонує ряд готових шаблонів, якими можна скористатися, щоб заощадити час і не створювати все з нуля.[5]

- ***Anki***

  програма для полегшення запам'ятовування слів, виразів і будь-який інший інформації за допомогою інтервальних повторень Основу бази даних Anki (колекції), що зберігається в форматі SQLite, становить список записів. Кожен запис є набором полів зі словами, визначеннями, вимова і т. П., З яких за вказаними в базі шаблонів автоматично створюються картки. Поля можуть складатися з HTML зі стилями, зображеннями, звуками, відео і LaTeX. Anki може робити так, щоб різні картки, з генеровані лише із одного запису (наприклад, «картинка» у слово» та «слово» у «картинка»), що не з'являлися поспіль. Доступна для користувача настройка і багатьох інших змінних. У випадку якщо при оцінюванні відповіді була допущена помилка, ця оцінка скасовується. Доступна статистика у вигляді цифр і графіків. Anki підтримує синхронізацію із серверною частиною,

реєстрація на якому не коштує грошей. При розміщенні «колоди» карток на серверній стороні, її можна синхронізувати з кількома пристроями або заучувати, використовуючи сайт (AnkiWeb), але для редагування і повторень рекомендується використовувати не сервер, а будь-яку з повнофункціональних версій.

- ***Memrise***

  Memrise - це навчальна платформа, створювана користувачами, яка використовує картки в якості інструменту навчання. Вона спеціалізується на вивченні мови, але також пропонує контент широкого спектру інших сфер. Memrise має більш ніж 150 мовних курсів на 25 мовах.

- ***SuperMemo***

  SuperMemo (від "Super Memory") - це метод навчання та програмний пакет, розроблений SuperMemo World та SuperMemo R&D з Piotr Woźniak у Польщі. Він ґрунтується на дослідженні довготривалої пам’яті і є практичним застосуванням методу дистанційного навчання з повторенням, який був запропонований для ефективного навчання рядом психологів ще в 30-х роках. На думку розробників SuperMemo та деяких інших прихильників дистанційного навчання повторення, процес може оптимізувати довгострокове здобуття знань. Програма SuperMemo зберігає базу запитань і відповідей, побудованих користувачем. Переглядаючи інформацію, збережену в базі даних, програма використовує алгоритм SuperMemo, щоб вирішити, які питання показати користувачеві. Потім користувач відповідає на запитання і оцінює своє відкликання - чи відповідали вони на питання легко, не вагаючись, зовсім не так і так далі - і їх рейтинг

використовується для підрахунку того, як швидко їм слід знову поставити запитання. Хоча точний алгоритм відрізняється від версії SuperMemo, загалом елементи, які важче запам'ятати, з’являються частіше. Окрім простих текстових запитань та відповідей, остання версія SuperMemo підтримує зображення, відео та HTML запитання та відповіді

- ***Mnemosyne***

  Mnemosyne - це програмне забезпечення з повторним повторенням. Розміщене повторення - це методика навчання, заснована на доказах, що показало, що збільшує швидкість запам'ятовування. Щодня програмне забезпечення відображає кожну карту, заплановану на повторення. Потім користувач оцінює свій спогад про відповідь картки за шкалою 0–5. Потім програмне забезпечення планує наступне повторення картки відповідно до рейтингу користувача цієї картки та бази даних карт в цілому. Це призводить до активного, а не пасивного процесу перегляду. Обґрунтуванням цього підходу є те, що (через ефект інтервалу) з часом кількість повторень, зроблених на день, зменшується, збільшуючи швидкість пригадування (порівняно з пасивними методами навчання), з мінімальним витраченим часом на навчання.

Таблиця. 1.1

Порівняльна таблиця популярних СДН іноземних мов

Система	Режим роботи	Розширюваність	Повністю бескоштовна
Moodle	online	Так	Ні
Anki	hybrid	Так	Так
Memrise	Hybrid	Ні	Ні
SuperMemo	offline	Ні	Так
Mnemosyne	offline	Так	Так

Слід визнати, що серед перерахованих, Anki є найбільш просунутою, та зручною системою, але вона також має свої певні обмеження, такі як – відсутність вбудованої можливість пошуку зображень, інтеграції з зовнішніми перекладачами та інше[6].

ВИСНОВКИ ДО РОЗДІЛУ 1

1. Аналіз недоліків та переваг існуючих СДН ноземних мов показав, що найбільш просуненою є Anki, проте вона також має певні обмеження.
2. За режимом роботи найпоширішими є гібридні системи, тобто системи, які можуть працювати і з Інтернетом і без його наявності.
3. Коротко розглянули особливості розробки СДН іноземних мов, а також оглядово познайомилися з різновидами СДН.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ ДИСТАНЦІЙНОГО НАВЧАННЯ ІНОЗЕМНОЇ МОВИ

Графічний інтерфейс є майже обов'язковою частиною будь-якої програми. Існують спеціальні архітектури, які дозволяють організувати ефективну взаємодію між бізнес-логикою та графічним інтерфейсом, вони дозволяють дуже гнучко розширювати фунціонал програми не впливаючи на існуючий. Однією із таких архітектурі є *MVC* (*Model View Controller*).

Також існують багато фреймворків, які полегшують програмування складних систем, в даній системі використовується Django для бекенд частини системи і VueJS для фронтенд. Крім цього, було вирішено додати систему контейнеризації Docker задля поліпшення процесу розробки та деплою. У якості сервера було використано Amazon Web Services, а в якості CI/CD – GitLab[7].

2.1 MVC

2.1.1 Загальна характеристика MVC

Модель-вид-контролер (англ. *Model-view-controller, MVC*) — це дуже популярний архітектурний шаблон, який може використовуватися під час проектування та розробки програмного забезпечення[8].

MVC поділяє систему на три частини декілька частин, а саме: модель даних, вигляд даних та керування, його основне застосування це інтерфейсу користувача від даних так, щоб зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача, та навпаки, зміни у інтерфейсі користувача майже не впливали на данні та роботу з ними..

2.1.2 Мета шаблону

Шаблон проектувався з метою покращення гнучкості дизайну та архітектури програм та систем, який має полегшувати розширення чи зміну програм, а також він має надавати можливість використання окремих компонент програми повторно. Окрім цього використання шаблону у

великих системах призводить до певної структурної впорядкованості та робить їх зрозумілішими завдяки зменшенню складності.

2.1.3 Структура MVC

Архітектурний шаблон *MVC*(або Модель-Вид-Контролер) поділяє програму на три частини. Вигляд (*View*) відповідальний за представлення даних користувачеві. Компонент Модель (*Model*) відповідає за зберігання цих даних та забезпечення інтерфейсу до них. А контролер (*Controller*) повинен керувати компонентами, отримувати сигнали у вигляді реакції на дії користувача, та повідомляти про зміни у Модель[9]. Ця внутрішня структура шаблону поділяє його на окремі незалежні частини: введення даних, обробку даних і виведення інформації. Модель - інкапсулює ядро даних і головний функціонал їх обробки, як це було відмічено вище. Разом з цим Модель не має залежати введення або виведення даних, за це відповідає компонент Вигляду, який крім цього може мати багато взаємопов'язаних областей, наприклад, поля форм і/або різні таблиці, у яких відображається дані. Функції Контролера – це моніторинг за подій, що виникають в результаті дій користувача (натиснення кнопки, зміна положення курсору миші або введення даних в текстове поле).

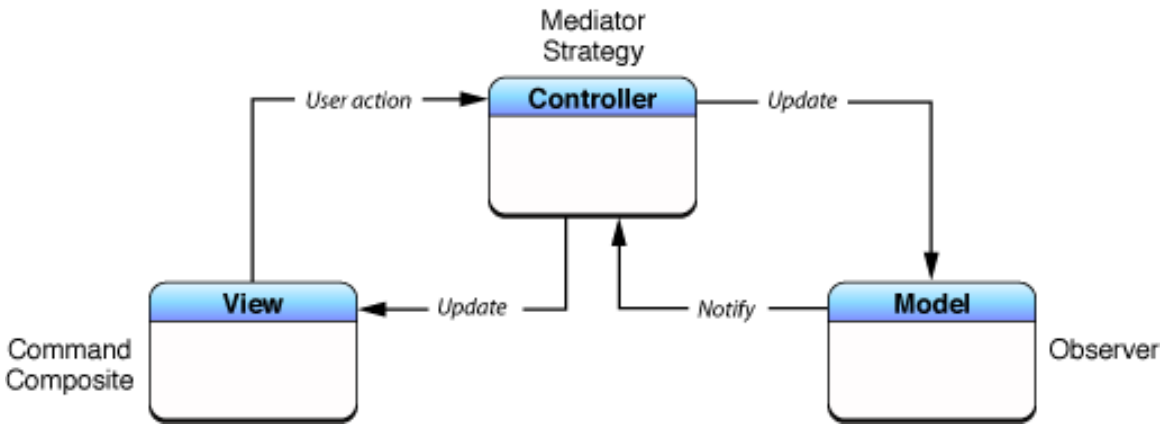


Рис. 2.1. Зв’язок між елементами шаблону

Події, що були зареєстровані - транслюються в запити різного вигляду, а далі ці запити йдуть до компоненти Моделі і/або об'єктам що відповідають за відображення даних(компонент Вигляду). Той факт, що модель відокремлена від вигляду дозволяє використовувати різні компоненти незалежно один від одного та незалежно від відображення інформації. Тобто, якщо користувач внесе зміни через Контролер до Моделі даних, то дані, подані візуальними компонентами, будуть автоматично відкориговані згідно до тих змін, що відбулися.

MVC дозволяє також змінювати реакцію виду на дії користувача. При цьому візуальне уявлення залишається колишнім. Наприклад, можна змінити реакцію на натискання клавіші або використовувати спливаючі меню замість командних клавіш. *MVC* інкапсулює механізм визначення реакції в об’єкті *Controller* [10]. Існує ієрархія класів контролерів, і це дозволяє без зусиль створити новий контролер як варіант вже існуючого.

Вид користується екземпляром класу, похідного від *Controller*, для реалізації конкретної стратегії реагування. Щоб реалізувати іншу стратегію, потрібно просто підставити інший контролер. Можна навіть замінити контролер виду під час виконання програми, змінивши тим самим реакцію на дії користувача. Наприклад, вигляд можна деактивувати, так що він взагалі не буде ні на що реагувати, якщо передати йому контролер, ігнорує події вводу.

Ставлення вид-контролер - це приклад патерну проектування стратегія. Стратегія - це об'єкт для представлення алгоритму[11]. Він корисний, коли ви хочете статично або динамічно підмінити один алгоритм іншим, якщо існує багато варіантів одного алгоритму або коли з алгоритмом пов'язані складні структури даних, які хотілося б інкапсулювати.

2.1.4 Шаблони програмування, що використовуються в MVC

Необхідно також відзначити, що як Вигляд, так і контролер можуть залежати від моделі. Однак модель ні в якому разі не повинна залежати ні від компоненту Вигляду, ні від контролера. Таке розподілення дозволяє

розробнику будувати модель незалежно від візуального представлення, крім цього – це також дозволяє створювати кілька різних Виглядів або Представлень для однієї та самої моделі.
Для реалізації схеми *MVC* використовується досить велика кількість шаблонів проектування (залежно від складності архітектурного рішення) такі, як фабричний метод, що дозволяє задати для виду клас контролера за замовчуванням, і декоратор для додавання до виду можливості прокрутки. Але основні відносини у схемі *MVC* описуються патернами спостерігач, компонувальник і стратегія.

MVC вперше був застосований при проектуванні Smalltalk - мови програмування, він був застосований як модель для інтерфейсу користувача. У сучасних технологіях концепція *MVC* представлена у python-фреймворку Django. Також раніше цей шаблон застосовували при реалізації каркаса Document-View.

У мові програмування Python *MVC* реалізовано на рівні стандартних класів та бібліотек. Використання MVC дає у розпорядження програмісту могутню структуру об’єктів-компонентів, у яких функції чітко розмежовані, що у свою чергу майже гарантує розширюваність та надійність розроблюваної системи.

2.1.5 Різновиди MVC

MVP

Даний підхід дозволяє створювати абстракцію подання. Для цього необхідно виділити інтерфейс подання з певним набором властивостей і методів. Презентер, в свою чергу, отримує посилання на реалізацію інтерфейсу, підписується на події уявлення та за запитом змінює модель.

Ознаки презентера:

- Двостороння комунікація з поданням;

- Представлення взаємодіє безпосередньо з презентери, шляхом виклику відповідних функцій або подій примірника презентера;
- Презентер взаємодіє з *View* шляхом використання спеціального інтерфейсу, реалізованого поданням;
- Один примірник презентера пов'язаний з одним відображенням.

MVVM

Даний підхід дозволяє зв'язувати елементи уявлення з властивостями і подіями *View*-моделі. Можна стверджувати, що кожен шар цього патерну не знає про існування іншого шару.

Ознаки *View*-моделі:

- Двостороння комунікація з поданням;
- *View*-модель - це абстракція подання. Зазвичай означає, що властивості уявлення збігаються з властивостями *View*-моделі / моделі;
- *View*-модель не має посилання на інтерфейс подання (*View*). Зміна стану *View*- моделі автоматично змінює уявлення і навпаки, оскільки використовується механізм скріплення даних (*Bindings*);
- Один примірник *View*-моделі пов'язаний з одним відображенням.

2.1.6 Переваги

Саме очевидне перевага, яку ми отримуємо від використання концепції *MVC* - це чіткий поділ логіки подання [12] (інтерфейсу користувача) і логіки програми. Підтримка різних типів користувачів, які використовують різні типи пристроїв є спільною проблемою наших днів. Наданий інтерфейс повинен відрізнятися, якщо запит приходить з персонального комп’ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність полягає в тому, що контролер вибирає різні види для виведення даних.

Крім ізолювання видів від логіки додатка, концепція *MVC* істотно зменшує складність великих додатків. Код виходить набагато більш

структурованим, і тим самим, полегшується підтримка, тестування та повторне використання рішень.

2.2 Огляд компонентів для Backend частини системи

2.2.1 Загальна характеристика фреймворку Django

Django - це веб-система Python високого рівня, яка заохочує швидкий розвиток та чистий, прагматичний дизайн. Побудований досвідченими розробниками, він піклується про багато клопот веб-розробки, тому ви можете зосередитись на написанні програми. Це безкоштовно та з відкритим кодом.

2.2.2 Можливості Django

Деякі можливості Django:

- підтримка багатьох мов
- гнучке API для доступу до БД у вигляді ORM
- загальні функції-шаблони або контролери
- кеш
- URL диспетчер, що реалізован на базі регулярних виразів
- Просунута система шаблонів та спадкування
- аутентифікація та авторизація, з підтримкою зовнішніх модулів аутентифікації: OpenID, LDAP, та інші.
- вбудована документація, яку можно генерувати автоматично
- архітектура додатків, що можна встановлювати у будь-які Django-сайти
- «middleware» - це система фільтрів, що використовується для побудови додаткових обробників користувацьких запитів
- Вбудований інтерфейс з локалізацією, що може використовуватися адміністраторами.

Деякі компоненти фреймворка між собою слабозв’язані, що дозволяє замінювати їх на альтернативи. Але з деякими (наприклад, з ORM) це зробити не дуже просто. Крім можливостей, вбудованих в ядро фреймворка, існують пакети, що розширюють його можливості.

На базі Django розроблено досить багато готових рішень, які розповсюджуються під вільною ліцензією, серед яких системи для управління інтернет-магазинами, універсальні системи управління вмістом, а також більш цілеспрямовані проекти.

2.2.3 Особливості Django

Django проектувався для роботи під управлінням Apache (з модулем mod_python) і з використанням PostgreSQL в якості бази даних. В даний час, крім PostgreSQL, Django може працювати з іншими СУБД: SQL Anywhere, Oracle, SQLite, Microsoft SQL Server, Firebird, DB2, MySQL та MariaDB. Django для роботи з СУБД надає власний ORM, в якому модель даних - це Python-класи, і по ним генерується структура бази даних[13].

Архітектура Django є дуже схожою на MVC. Класичний MVC-контролер приблизно відповідає рівню, який в Django називається Представлення (View), а логіка Вигляду в Django реалізується рівнем Шаблонів (Templates), саме із-за цього архітектуру Django часто називають MTV(Модель-Шаблон- Представлення).

Спочатку розробка Django велася для забезпечення більш зручної роботи з новинними ресурсами, що досить сильно відбилося на архітектурі: фреймворк надає низку методів, які значно прискорюють розробку веб-сайтів. Наприклад, в Django вбудовано додаток для керування сутньостями , саме тому розробнику вже не потрібно самому створювати контролери та сторінки для адміністративної частини сайту, бо вони вже готові. Цей додаток можна підключити на будь-який Django-сайт. Цей додаток дозволяє видаляти, створювати

та змінювати довільні об'єкти наповнення сайту, крім цього Django зберігає історію усіх скоєних дій, а ще, Django надає гнучкий інтерфейс для управління користувачами та користувацькими групами.

Django використовують дуже багато компаній, з усіх них можна перелічити декілька найбільших, це такі веб-сайти як як Pinterest, Disqus, The Washington Times, Instagram, Mozilla, lamoda і ін.

2.3 Огляд компонентів для Frontend частини системи

HTML (мова розмітки гіпертексту) та CSS (каскадні таблиці стилів) - дві основні технології для створення веб-сторінок. HTML надає структуру сторінки, CSS (візуальну та слухову) макет для різних пристроїв. Поряд із графікою та сценаріями, HTML та CSS є основою для створення веб-сторінок та веб-додатків.

HTML - це мова для опису структури веб-сторінок. CSS - це мова для опису презентації веб сторінок, включаючи кольори, макет та шрифти. Це дозволяє адаптувати презентацію до різних типів пристроїв, таких як великі екрани, маленькі екрани чи принтери. CSS не залежить від HTML і може використовуватися з будь-якою мовою розмітки на основі XML. Відокремлення HTML від CSS полегшує підтримку сайтів, обмін таблицями стилів на сторінках та адаптування сторінок для різних середовищ. Це називається відокремленням структур від викладу.

Bootstrap - це величезна колекція зручних, багаторазових частин коду, написаних у HTML, CSS та JavaScript. Це також фреймворк для розробки клієнтського інтерфейсу для користувачів, цей набір дозволяє розробникам швидко створювати повністю адаптивні веб-сайти. Bootstrap легко дозволяє створювати сітки, має багато компонентів які можна кастомізувати, а також має багату документацію[14].

Vue.js - це фреймворк з відкритим кодом JavaScript MVVM JavaScript для створення інтерфейсів користувача та односторінкових програм. Vue.js поступово адаптується під архітектуру, яка зосереджена на декларативному візуалізації та складі компонентів. Основна бібліотека орієнтована лише на

шару відображення. Розширені функції, необхідні для складних додатків, таких як маршрутизація, управління державою та інструменти побудови, пропонуються через офіційно підтримувані бібліотеки та пакети, а Nuxt.js - одне з найпопулярніших рішень / Vue.js дозволяє розширювати HTML за допомогою атрибутів HTML, що називаються директивами. Директиви пропонують функціональні можливості для програм HTML і надходять як вбудовані, так і визначені користувачем директиви[15].

Компоненти Vue розширюють основні елементи HTML для інкапсуляції коду багаторазового використання. На високому рівні компоненти - це спеціальні елементи, до яких компілятор Vue приєднує поведінку. У Vue компонент, по суті, є екземпляром Vue з попередньо визначеними параметрами. Фрагмент коду нижче містить приклад компонента Vue. Компонент представляє кнопку і друкує кількість натискань на кнопку.

Vue використовує синтаксис шаблонів на основі HTML, який дозволяє прив'язувати наданий DOM до базових даних екземпляра Vue. Усі шаблони Vue є дійсним HTML, який можна проаналізувати за допомогою веб-переглядачів та HTML-аналізаторів. Vue збирає шаблони у функції віртуального візуалізації DOM. Віртуальна модель об'єкта документа (або "DOM") дозволяє Vue відтворювати компоненти в своїй пам'яті перед оновленням браузера. У поєднанні із системою реактивності Vue здатний обчислити мінімальну кількість компонентів для повторного відображення та застосування мінімальної кількості маніпуляцій DOM, коли стан додатка змінюється[11].

Vue має систему реактивності, яка використовує звичайні об’єкти JavaScript та оптимізує повторну візуалізацію. Кожен компонент відслідковує свої реактивні залежності під час його візуалізації, тому система точно знає, коли потрібно відредагувати, та які компоненти повторно візуалізувати.

Традиційним недоліком односторінкових програм (SPA) є неможливість ділитися посиланнями на

точну "під" сторінку в межах конкретної веб-сторінки. Оскільки SPA-сервіси обслуговують своїх користувачів лише одну відповідь на основі URL-адреси від сервера (як правило, він пропонує index.html або index.vue), закладки на певні екрани або обмін посиланнями на конкретні розділи, як правило, важко, якщо не неможливо. Щоб вирішити цю проблему, багато маршрутизатори на стороні клієнта розмежовують свої динамічні URL-адреси "hashbang" (#!), Наприклад page.com/#!/. Однак у HTML5 більшість сучасних браузерів підтримують маршрутизацію без хеш-багів.

Vue надає інтерфейс для зміни того, що відображається на сторінці, залежно від поточного шляху до URL-адреси - незалежно від того, як це було змінено (чи за посиланням електронною поштою, оновленням або на сторінках). Крім того, використання маршрутизатора на передньому кінці дозволяє навмисно переходити шлях браузера, коли певні події браузера (тобто кліки) відбуваються на кнопках або посиланнях. Сам Vue не має маршрутизованих хешованих маршрутизацій. Але пакет «vue-router» з відкритим кодом надає API для оновлення URL-адреси програми, підтримує кнопку повернення (історія навігації) та скидає пароль електронної пошти або посилання для підтвердження електронної пошти з параметрами URL-адреси аутентифікації. Він підтримує відображення вкладених маршрутів до вкладених компонентів і пропонує тонкозернистий контроль переходу. З Vue розробники вже складають програми з невеликими будівельними блоками, що будують більші компоненти. Якщо в суміш додано vue router, компоненти повинні бути просто відображені до маршрутів, до яких вони належать, а батьківські / кореневі маршрути повинні вказувати, куди діти повинні переводитись.

2.4 Компонент Docker для розгортки додатку

Docker - використовує віртуалізацію рівня ОС для доставки програмного забезпечення в пакетах, званих контейнерами. Контейнери відокремлені один від одного та поєднують власне програмне забезпечення,

бібліотеки та файли конфігурації; вони можуть спілкуватися між собою через чітко визначені канали. Усі контейнери управляються одним ядром операційної системи і тому використовують менше ресурсів, ніж віртуальні машини.

Docker може упакувати додаток та його залежності у віртуальний контейнер, який може працювати на будь-якому сервері Linux. Це допомагає забезпечити гнучкість та портативність, що дозволяє запускати програму в різних місцях, незалежно від локальних даних, у відкритій хмарі чи в приватній хмарі. Docker використовує функції виділення ресурсів ядра Linux (наприклад, групи груп та простори імен ядра) та файлову систему, сумісну з об'єднанням (наприклад, OverlayFS), щоб дозволити контейнерам запускатись в одному екземплярі Linux, уникаючи накладних витрат на запуск та підтримка віртуальних машин. Оскільки контейнери Docker мають невелику вагу, один сервер або віртуальна машина можуть запускати кілька контейнерів одночасно. Типовий випадок використання Докера передбачає запуск восьми контейнерів на хоста, але що чверть проаналізованих організацій працює 18 або більше на одного хоста.

Підтримка ядра Linux для просторів імен здебільшого виокремлює погляд програми на робоче середовище, включаючи дерева процесів, мережу, ідентифікатори користувачів та встановлені файлові системи, тоді як групи ядер забезпечують обмеження ресурсів для пам'яті та процесора. Починаючи з версії 0.9, Docker включає власний компонент (званий "libcontainer") для прямого використання засобів віртуалізації, що надаються ядром Linux, на додаток до використання абстрагованих інтерфейсів віртуалізації через libvirt, LXC та systemd-nspawn.

Docker реалізує API високого рівня, щоб забезпечити легкі контейнери, які запускають процеси ізольовано.

Програмне забезпечення Docker складається з трьох компонентів:

- Демон Docker, званий dockerd, - це стійкий процес, який управляє контейнерами Docker і обробляє контейнерні об'єкти. Демон слухає

запити, надіслані через API Docker Engine. Клієнтська програма Docker, що називається docker, забезпечує інтерфейс командного рядка, який дозволяє користувачам взаємодіяти з демонами Docker.

- Докерські об'єкти - це різні об'єкти, які використовуються для збирання програми в Докер. Основні класи об’єктів Docker - образи, контейнери та сервіси. Контейнер Docker - це стандартизоване, інкапсульоване середовище, яке запускає програми. Керується контейнером за допомогою API Docker або CLI. Образи Докера - це шаблон лише для читання, який використовується для створення контейнерів. Образи використовуються для зберігання та доставки програм. Сервіси Docker дозволяє масштабувати контейнери через кілька демонів Docker. Результат відомий як рій, набір демонів, що співпрацюють, які спілкуються через API Докера.
- Реєстр Docker - це сховище для зображень Docker. Клієнти Docker підключаються до реєстрів, щоб завантажити зображення для використання або завантажити створені ними зображення. Реєстри можуть бути державними або приватними. Два основні державні реєстри - Docker Hub та Docker Cloud. Docker Hub - це реєстр за замовчуванням, де Docker шукає зображення. Реєстри докерів також дозволяють створювати сповіщення на основі подій.

2.5 CI/CD та інше

Окрім вище перерахованих технологій також було використано ще декілька корисних сервисівя, а саме – GitLab CI, PyTest, AWS, ElasticSearch, Redis.

Це було зроблено для покращення якості додатку, швидкого процесу його розгортування, пошуку, а також тестування. Нижче приведено опис цих технологій.

2.5.1 GitLab CI/CD

Для розробки системи також було зроблено настройку конвеєру доставки програмного забезпечення, так званого CI/CD.

Конвеєр доставки програмного забезпечення забезпечує швидкі, автоматизовані та відтворювані випуски вихідного коду. Загальна конструкція того, як це робиться, називається "безперервною доставкою". Процес, який починається з конвеєра, називається "безперервною інтеграцією". Процес, що забезпечує якість, називається "безперервним тестуванням", а процес, який робить кінцевий продукт доступним для користувачів, називається "безперервне розгортання".
Безперервне використовується для опису багатьох різних процесів, які слідують практиці, яку я тут описую. Це не означає "завжди бігати". Це означає "завжди готовий до запуску". У контексті створення програмного забезпечення воно також включає декілька основних концепцій найкращих практик. Це:

- Часті випуски. Завдання постійних практик полягає в тому, щоб забезпечити доставку якісного програмного забезпечення з частими інтервалами. Частота тут мінлива і може бути визначена командою або компанією. Для деяких продуктів один раз на квартал, місяць, тиждень або день може бути досить частим. Для інших може бути бажаним і виконаним кілька разів на день. Безперервне також може приймати аспект "епізодичного, за потреби". Кінцева мета та сама: доставити якісні оновлення програмного забезпечення кінцевим користувачам у повторюваному надійному процесі. Часто це може бути зроблено з незначною взаємодією або навіть незнанням користувачів.
- Автоматизовані процеси: Ключовою частиною включення цієї частоти є автоматизовані процеси для обробки майже всіх аспектів виробництва програмного забезпечення. Сюди входить побудова, тестування, аналіз, версія та, в деяких випадках, розгортання.

- Повторюваність: Якщо ми використовуємо автоматизовані процеси, які завжди мають однакову поведінку з однаковими входами, то обробка повинна повторюватися. Тобто, якщо ми повернемось назад і введемо ту саму версію коду, що і вхід, ми повинні отримати той самий набір результатів. Це також передбачає, що у нас є ті самі версії зовнішніх залежностей (тобто інші результати, які ми не створюємо, які використовує наш код). В ідеалі це також означає, що процеси в наших трубопроводах можна переосмислити і відтворити.
- Швидка обробка: "Швидка" тут відносний термін, але незалежно від частоти оновлень випусків програмного забезпечення, очікується, що безперервні процеси ефективно обробляють зміни від вихідного коду до результатів. Значна частина цього переймається автоматизацією, але автоматичні процеси все ще можуть бути повільними. Наприклад, інтегроване тестування в усіх аспектах продукту, яке займає більшу частину дня, може бути занадто повільним для оновлень продукту, які мають новий реліз кандидатів кілька разів на день.

У якості CI/CD було обрано GitLab CI завдяки наступним перевагам:

- CI / CD GitLab є частиною GitLab, що дозволяє проводити одну розмову від планування до розгортання
- Відкритий код: CI / CD є частиною як відкритого джерела GitLab Community Edition, так і фірмового GitLab Enterprise Edition
- Масштабованість: тести виконуються розподіленими на окремих машинах, до яких можна додати скільки завгодно
- Оптимізовано для доставки: кілька етапів, ручні ворота розгортання, середовища та змінні

Крім цього, GitLab - це одна з програм для всього життєвого циклу DevOps, а саме: збірки додатку, запуску тестів, ревью коду, деплою та моніторингу статусу додатка.

2.5.2 PyTest

Це надійний інструмент тестування Python, pytest може використовуватися для всіх типів і рівнів тестування програмного забезпечення. Pytest може бути використаний командами розвитку, командами з забезпечення якості, незалежними групами тестування, особами, які практикують TDD та проектами з відкритим кодом. Насправді, pytest пропонує потужні функції, такі як перезапис «assert», сторонню модель плагінів та потужну, але просту модель кріплення, яка не має собі рівних у будь-яких інших тестових рамках.

У проекті pytest був використаний для тестуваня бекенд частини проекту, особливо бізнес-логіки.

2.5.3 ElasticSearch

Для пошуку рейтингу словників серед усіх користувачів в системі був обраний ElasticSearch. Це дозволило зробити пошук по великому обсягу даних дуже швидким.

ElasticSearch - це розподілений, відкритий движок для пошуку та аналітики для всіх типів даних, включаючи текстові, числові, геопросторові, структуровані та неструктуровані. Elasticsearch побудований на Lucene Apache і вперше був випущений у 2010 році Elasticsearch N.V. Відомий своїм простим REST API, розподіленою природою, швидкістю та масштабованістю, Elasticsearch є центральним компонентом Elastic Stack - набору інструментів з відкритим кодом для прийому даних, збагачення, зберігання, аналізу та візуалізації. Elasticsearch використовується для різних типів пошуку, логування, метрік та моніторінгу, моніторінгу продуктивності, а також для аналітики.

2.5.4 Redis

У деяких випадках було використано кешування даних за допомогою Redis.

Redis - це сховище з відкритим кодом для зберігання структур даних в пам'яті, який використовується як база даних, кеш-пам'ять та брокер повідомлень. Він підтримує структури даних, такі як рядки, хеші, списки, набори, відсортовані набори з діапазонами запитів, растрових зображень, гіперлогів, геопросторових індексів з радіусними запитами та потоками. Redis має вбудовану реплікацію, Lua, транзакції та різні рівні стійкості на диску, а також забезпечує високу доступність через Redis Sentinel та автоматичне розділення з кластером Redis.

Щоб досягти своїх видатних показників, Redis працює з набором даних пам'яті. Залежно від випадку використання, ви можете зберігати його або скидаючи набір даних на диск раз у раз, або додаючи кожну команду до журналу. Наполегливість може бути відключена, якщо вам просто потрібен багатофункціональний керований кеш пам'яті.

2.5.5 Amazon Web Services (AWS)

Amazon Web Services (AWS) надає платформи та API для хмарних обчислень на замовлення фізичним особам, компаніям та урядам на основі дозованої оплати. У сукупності ці веб-сервіси хмарних обчислень забезпечують набір примітивної абстрактної технічної інфраструктури та розподілених будівельних блоків та інструментів. Один з таких сервісів - Amazon Elastic Compute Cloud (EC2), який дозволяє користувачам мати в своєму розпорядженні віртуальну кластерну мережу комп'ютерів, доступну весь час через Інтернет. Версія віртуальних комп’ютерів AWS емулює більшість атрибутів реального комп'ютера, включаючи апаратні центральні процесорні блоки (процесори) та графічні одиниці обробки (GPU) для обробки; локальна / оперативна пам'ять; накопичувач на жорсткому диску / SSD; вибір операційних систем; мережа; попередньо завантажене прикладне

програмне забезпечення, таке як веб-сервери, бази даних та управління відносинами з клієнтами (CRM).

Технологія AWS впроваджена на серверних фермах у всьому світі. AWS відокремлено на різні служби; кожен може бути налаштований по-різному залежно від потреб користувача. Користувачі повинні мати можливість бачити параметри конфігурації та окремі серверні карти для послуги AWS.

Понад 100 послуг складають портфоліо Amazon Web Services, включаючи послуги з обчислень, баз даних, управління інфраструктурою, розробкою програм та безпекою.

ВИСНОВКИ ДО РОЗДІЛУ 2

Матеріал, що розглянутий у розділі 2 дозволяє сформувати задачі та план до розробки запропонованої системи.

Виходячи з огляду обраних компонентів, можна перерахувати наступні властивості системи закладені на етапі проектування:

- Автоматичне розгортка;
- Автоматичне тестування коду за допомогою модуля pytest;
- Збірка всієї системи в docker-контейнери;
- Міграції бази даних;
- Розгортка додатку у AWS;
- Адаптивний веб-інтерфейс для клієнта;
- Адаптивний веб-інтерфейс для адміністратора;
- Збереження даних у кеш
- Швидкий пошук

РОЗДІЛ 3
ОПИС АРХІТЕКТУРИ РОЗРОБЛЮВАНОЇ СИСТЕМИ

Архітектура системи полягає у розробці backend та frontend частин проекту, які полягають у реалізації панелі користувача та панелі адміністратора, при цьому клієнт має доступ лише до своїх словників та слів, в той час як адміністратор має доступ до даних усіх користувачів для того щоб здійснювати дії модератора. Також frontend має звертатися до backend'у за допомогою API (Application Programming Interface). В якості бази даних використано реляційну система управління базами даних - Postgres. Для написання API обрано REST (Representational State Transfer) структуру взаємодії backend та frontend. В якості frontend виступає фреймворк VueJS, який використовується для клієнта та адміністратора.

Захищеність даних системи

Захист даних здійснено тим, щ є аутентифікація яка зроблена за допомогою JSON Web Tokens (JWT), при цьому токен це способом авторизації для кожного запиту від клієнта до сервера. Токени генеруються на сервері на базі спеціально обраного ключа, цей токен зберігається на клієнті та може використовується при авторизації будь-якого запиту. Даний токен генерується на сервері тільки після вдалого запиту на аутентифікацію – введення логіну і паролю. І порівняння хешованого пароля та логіна на сервері з відповідними даними сутності User у базі даних повернуть згенерований токен у відповідь.

Надалі усі API запити вимагають попередньої авторизації для звернення до них. Кожний користувач може мати декілька прав у системі: Адміністратор (Admin) та Користувач (User). Кожна роль передбачає різні права доступу до системи та різний функціонал у системі. Адміністратор має доступ до даних усіх користувачів, він може видаляти, добавляти та редагувати абсолютно будь-яку інформацію.

Користувач же може створювати переклади лише в межах свого профілю.

Модель бази даних

При розробці структури бази даних було враховано усі дані і параметри, які важливі для зберігання і їх подальшого аналізу.

На рис. 3.1. зображено частину моделі бази даних частини для користувача розроблюваної системи.

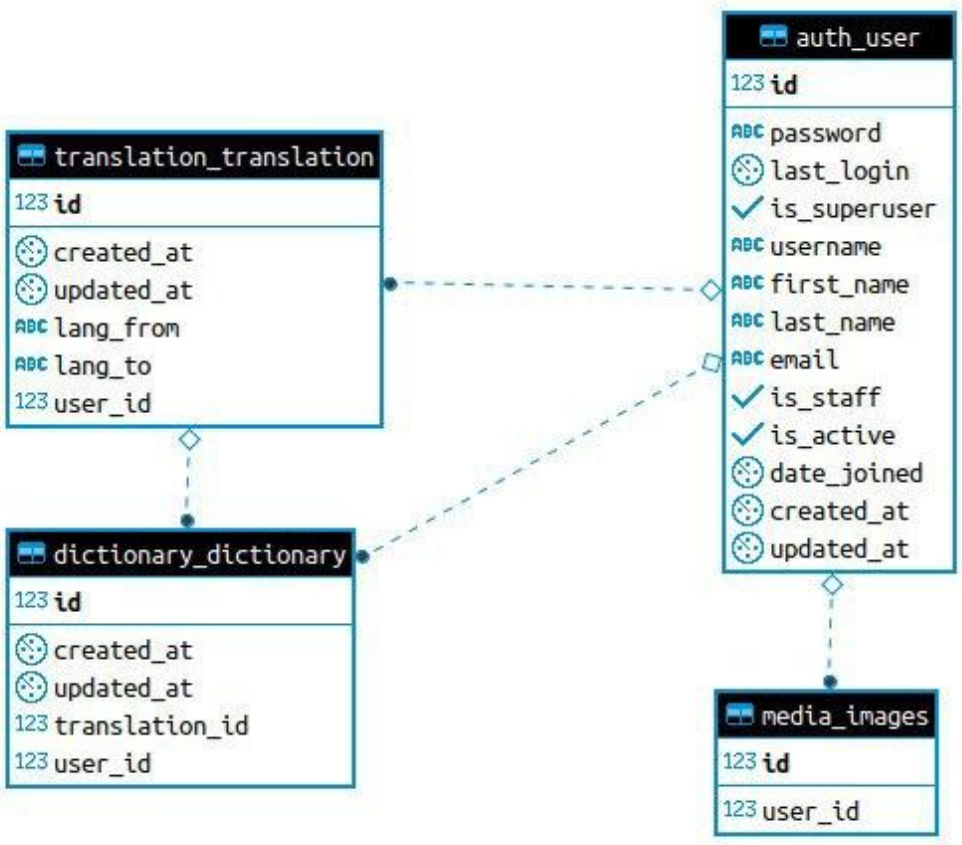


Рис. 3.1. Частина моделі бази даних частини

На рис. 3.2 зображено загальну схему взаємодії усіх компонентів системи.

Клієнт запрошує дані з бекенду, той доступається або до кешу, або до базі даних, або до системи пошуку ElasticSearch, як тільки бекенд отримує необхідні дані, вони відразу же передаються на фронтенд.

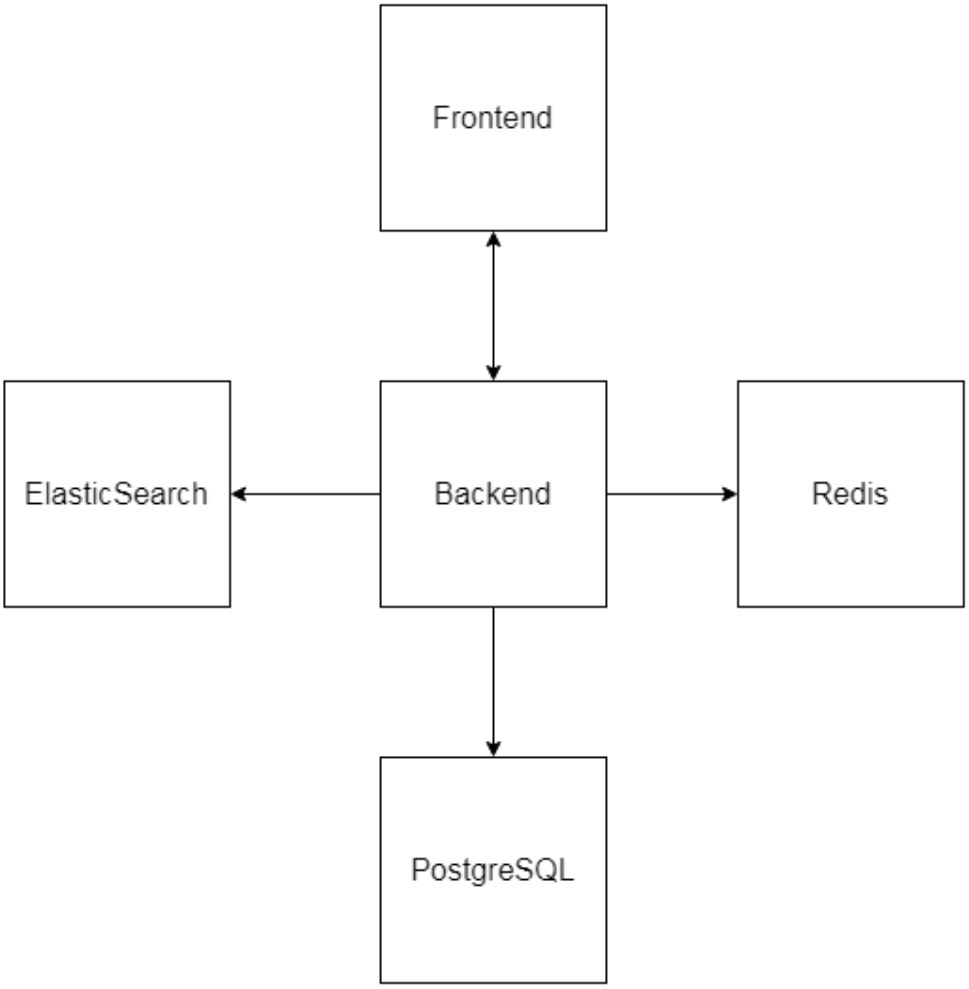


Рис. 3.2 Загальна схема взаємодії усіх компонентів системи

Переклад має зовнішній зв'язок із словником, також він має деякі свої специфічні поля та методи.

Користувач може мати різні права, тому для цього є спеціальне поле в якому це вказується. Також треба сказати, що на сутність користувача зав’язана сутність словник та переклад.

3.1 Налаштування середовища для розробки системи

Для швидкої розробки на локальній системі треба встановити сам docker та docker-compose, який дозволяє легко описувати запуск усіх контейнерів, а також те, як вони взаємодіють.

Для postgres додатково потрібно вказати логін та пароль користувача.

Налаштування Docker контейнерів:

```
services:
  base-app: &base-app
    build:
      context: .
      dockerfile: config/docker/Dockerfile.application
    image: lexicone-app
    read_only: true
    tty: true
    volumes:
      - .:/work:delegated



  base-frontend: &base-frontend
    build:
      context: .
      dockerfile: config/docker/Dockerfile.frontend
    image: lexicone-frontend
    volumes:
      - .:/work:delegated
      - /work/node_modules

  django-app:
    ports:
      - 8000:8000
    depends_on:
      - redis
    command: python3 manage.py runserver 0.0.0.0:8000 --insecure


  frontend:
    <<: *base-frontend
    ports:
      - 3000:3000
    tty: true
  command: sh -c "yarn start"
```

```
elasticsearch:
  image: elasticsearch:5.6.16
  tmpfs: /usr/share/elasticsearch/data

redis:
  build: &build-redis
    context: .
    dockerfile: config/docker/Dockerfile.redis


 postgres:
   build:
     context: .
     dockerfile: config/docker/Dockerfile.postgres
   environment:
     POSTGRES_USER: postgres
     POSTGRES_PASSWORD: postgres
     PGDATA: /var/lib/postgresql/data/pgdata
```

Опція context вказує на контекст де буде відбуватися збірка образу, опція dockerfile вказує на шлях до docker файлу образу.

Також для деяких контейнерів потрібно зробити мапінг портів на хостову машину через опцію port.

Compose має команди для управління всім життєвим циклом програми:

- Запуск, зупинка та відновлення послуг
- Перегляд стану запущених служб
- Потокове виведення журналу запущених служб
- Запустіть разову команду на сервісі

Конфігурація GitLab:

```
variables:
  CONTAINER_TAG: ${CI_COMMIT_REF_SLUG}_${CI_COMMIT_SHORT_SHA}
  COMPOSE: docker-compose -p ${CI_COMMIT_SHA} -f docker-compose.test.yml
  PROJECT_NAME: lexicone


stages:
  - manual
  - lint-test
  - build-js
  - build
  - deploy



.lint-test:
  stage: lint-test
  image: $CI_REGISTRY_IMAGE/utils:latest


lint-server:
  extends: .lint-test
  script:
    - (cd lexicone && flake8 --config=.flake8 . )
    - mypy --config-file lexicone/mypy.ini lexicone


deploy-prod:
  stage: deploy
  only:
    - master
  script:
    - aws push


build-utils:
  stage: manual
  when: manual
  script:
    - docker pull $CI_REGISTRY_IMAGE/utils:latest || true
    - docker build --network=host --cache-from $CI_REGISTRY_IMAGE/utils:latest -t
$CI_REGISTRY_IMAGE/utils:latest -f Dockerfile.utils .
    - docker push $CI_REGISTRY_IMAGE/utils:latest



lint-front:
  extends: .lint-test
  cache:
    key: ${CI_COMMIT_REF_SLUG}
    paths:
      - front/node_modules/
  before_script:
    - (cd front && npm install)
  script:
    - echo ${CI_COMMIT_REF_SLUG}
    - (cd front && npm run lint)



build-js-artifact:
  stage: build-js
  image: $CI_REGISTRY_IMAGE/utils:latest
  script:
    - cd front
    - npm install
    - npm run build
  artifacts:
    paths:
      - lexicone/dist
    expire_in: 1 day
```

```
test:
  extends: .lint-test
  script:
    - $COMPOSE build testserver
    - $COMPOSE up -d testdb
    - $COMPOSE run check_testdb
    - $COMPOSE run --rm testserver sh -c "cd lexicone && alembic upgrade head"
    - $COMPOSE run --rm testserver sh -c "pytest --disable-warnings --doctest-modules --
ignore=lexicone/migrations/ --cov=lexicone --cov-config=.coveragerc --cov-report term-
missing lexicone/"
  after_script:
    - $COMPOSE down -v
```

3.2 Розробка backend частини системи

При проектуванні системи було прийнято рішення створення REST структури з CRUD ендпоїнтами для видалення, отримання та оновлення, і створення даних для їх запису у таблицю сутностей.

Далі треба забезпечити цю структуру захищеним доступом до даних, тобто треба вказати рівень доступу до певних ендпоїнтів, що вказано на рис. 3.3.

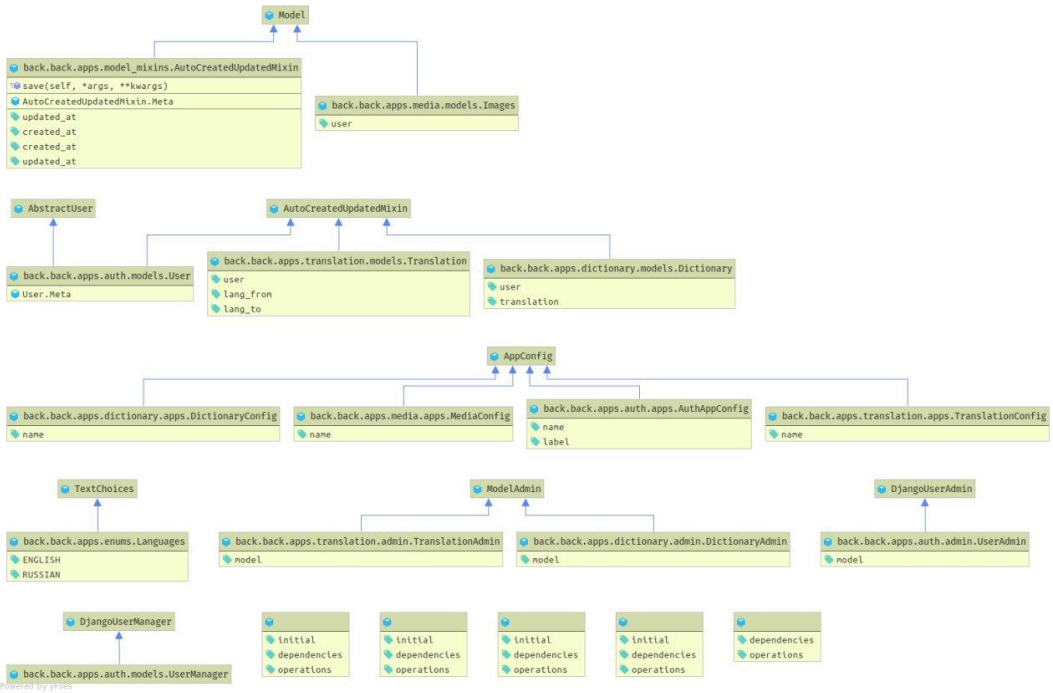


Рис. 3.3. Діаграма класів

3.3 Розробки frontend частини системи

Структуру директорій було зроблено, так як показано на рис.3.3

/api – зберігається код який відноситься до взаємодії з сервером.

/assets – містить у собі усі артефакти які потрібні для інтерфейсу користувача, а саме, переважно це картинки у різних форматах

/plugins – інтеграція із зовнішніми плагінами

/router – опис роутінгу на кліенті

/scss - стилі

/store – сховище даних, сюди дані потраплять з бекенду, та звідси вони їх забирають, для відображення даних

/views – безпосередньо сторінки додатку, з усією логікою та іншим.

/mixins – домішки, це код який не використовується у багатьох частинах проекту

3.4 Розробка панелі адміністратора

Для керування даними усіх користувачів треба створити спеціальний інтерфейс управління до певних сутностей, які мають важливу бізнес-логіку. Такими сутностями є переклад, користувач, словники..

Важливою частиною побудови будь-якого інтерфейсу є структура авторизації користувача та організація прав доступу до певного компоненту..

Структуру для керування сутностями розміщено у спеціальному модулю, який має бути доступний усім компонентам системи, що дозволяє використовувати інтерфейси сутності для їх керування в усіх сервісах та компонентах системи.

Кожен компонент для керування певною сутністю складається з декількох компонентів – для отримання списку записів даної сутності, детального перегляду полів запису, модифікація або створення нового запису. При цьому важливим компонентом є налаштування шляхів і конфігурація файлу для переліку усіх компонентів модуля певної сутності, конфігурація сервісу для запитів до серверу. Кожен з компонентів відповідає певному CRUD ендпоїнту серверу, що також підтверджує один з принципів структури SOLID.

Для отримання усіх записів сутності створено таблицю керування кожним записом, що дозволяє отримати загальну інформацію про кожну сутність та продовжити роботу з певним записом – видалити, переглянути детальну інформацію або редагувати, що зображено на рис. 4.18.

ВИСНОВКИ ДО РОЗДІЛУ 3

Створено структуру для системи дистанційного навчання іноземної мов.

У розділі 3 розроблено структуру та взаємодію між елементами системи: клієнтським кодом, серверним кодом, базою даних та кешем.

Сервер дістає із бази даних дані, що запрошує клієнт через протокол HTTP. Також було виконано збереження користувацьких даних у базу даних за відповідним ідентифікатором користувача. Інтерфейс адміністратора дозволяє керувати даними усіх користувачів. Також був реалізовано пошук, який дозволяє користувачам швидко знайти потрібні дані.

Для збереження фото та аудіо було налаштовано Amazon S3 технології, що дозволяє незалежно звертатись до файлу, який зберігається у спеціальному сховищі даних, також було реалізовано відповідний інтерфейс з серверного боку для доступу к S3.

Створено REST API за допомогою фреймворку Django, цей інтерфейс використовує клієнт, який реалізоване за допомогою сучасного JavaScript фреймворку – VueJS.

РОЗДІЛ 4
ІНСТРУКЦІЯ ДЛЯ КОРИСТУВАЧА

4.1 Інструкція для користувача

Першим етапом користування системою є створення профілю. Для створення профілю необхідно ввести ім’я, пошту та пароль.

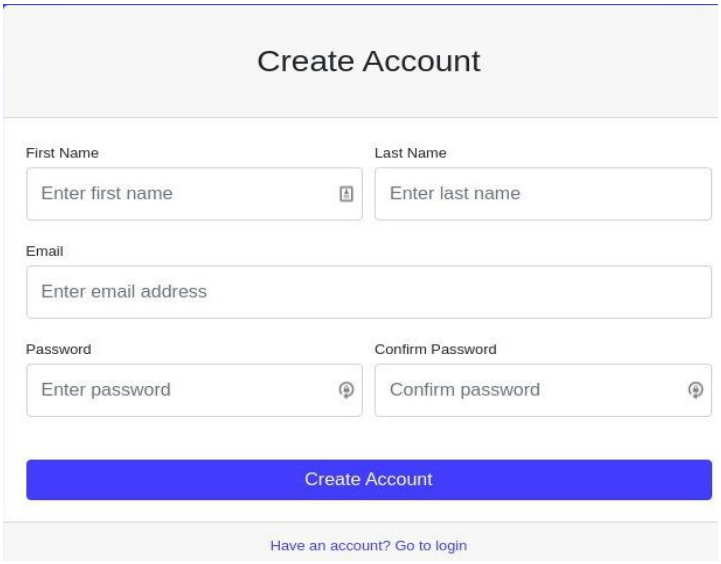


Рис. 4.1. Реєстрація

Після реєстрації користувач може ввійти у систему по вказаній пошті та паролю.

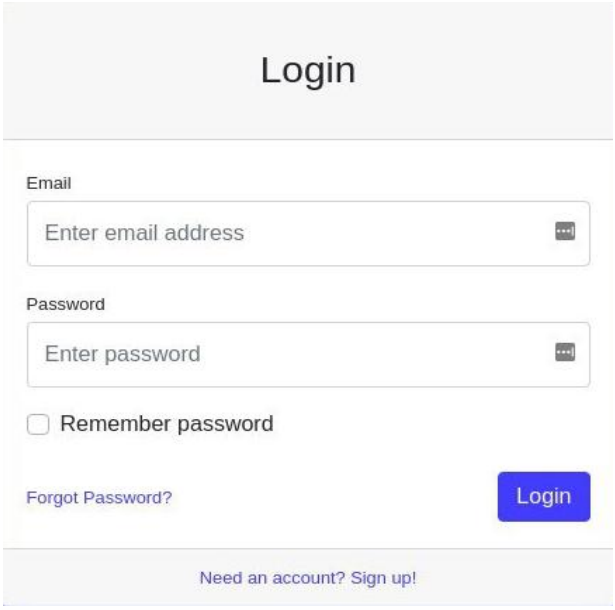


Рис. 4.2. Вхід

У випадку коли користувач намагається доступитися до сторінки до якої в нього немає доступу він отримає помилку із кодом 401.

Рис. 4.3. Помилка 401 - не авторизовано.

У разі випадку коли користувач забув свій пароль він може його відновити натиснувши на відповідну кнопку, форма, що відкриється зображена на наступному рисунку.

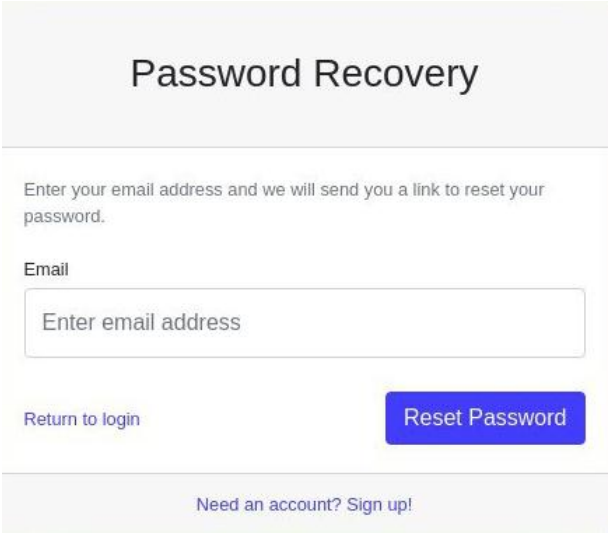


Рис. 4.4. Форма відновлення паролю

Якщо користувач закінчив роботу з системою він може вийти натиснувши на відповідну кнопку у верхньому лівому куті.

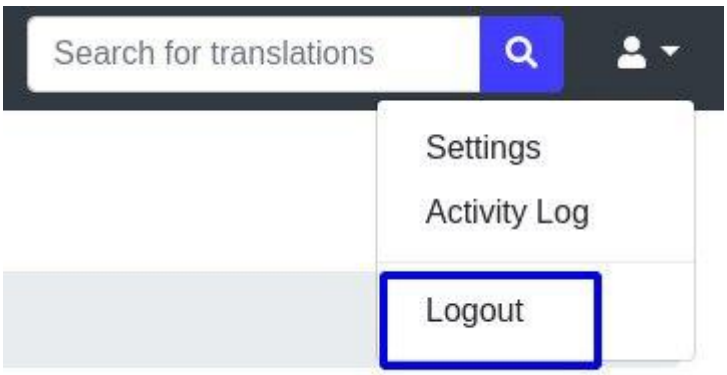


Рис. 4.5 Вихід із системи

Також є можливість подивитися довідкові дані про іншого користувача системи натиснувши на його іконку, дані що відобразяться зображені на наступному рисунку.

Рис. 4.6. Перегляд профілю іншого користувача.

У випадку коли користувач у адресну стрічку вписав неправильний запит, то тоді він побачить відповідну помилку із статус кодом – 404.

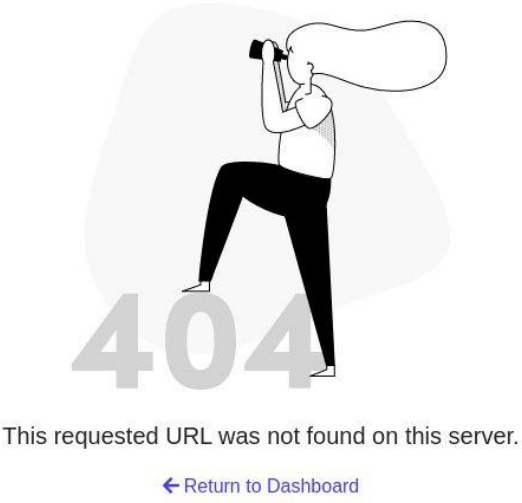


Рис. 4.7. Помилка 404 – неправильний запит.

Форми пошуку готових словників та перекладів відображено у вигляді двох текстових полів.

При пошуку буде здійснюватися запит в ElasticSearch. Також треба відмітити, що дані можуть бути закешовані у Redis. Після натисканні на кнопку пошуку користувач отримає результат.

Рис. 4.8. Форми для пошуку словників та перекладів

Також в користувача є можливість переглянути свою статистику вивчення слів:

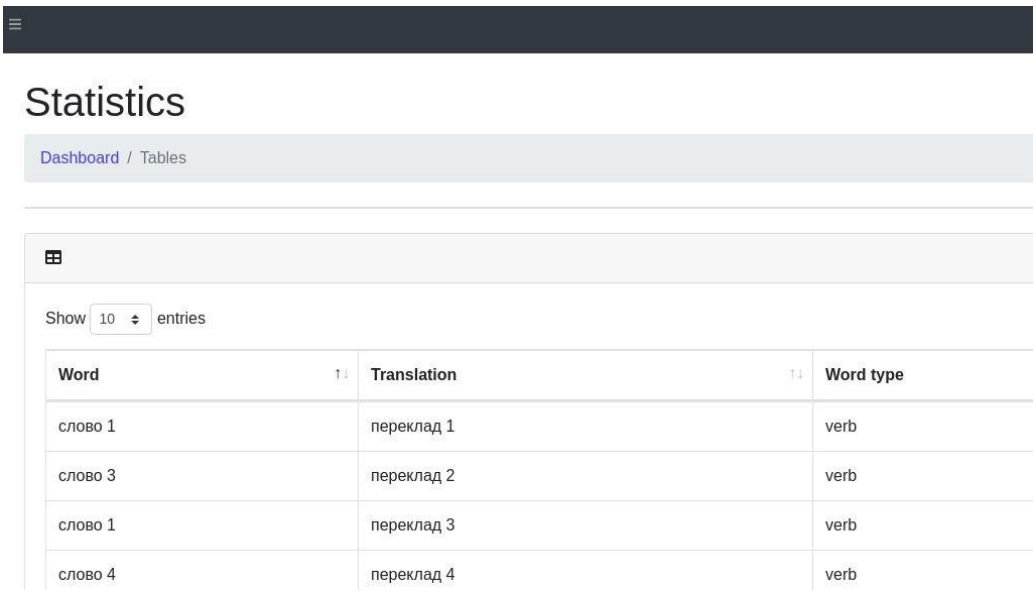


Рис. 4.9. Статистика вивчення слів

Якщо потрібно, користувач може змінити особисті контактні дані, натиснувши на кнопку «Account», після чого відкриється відповідна форма яку можна редагувати:

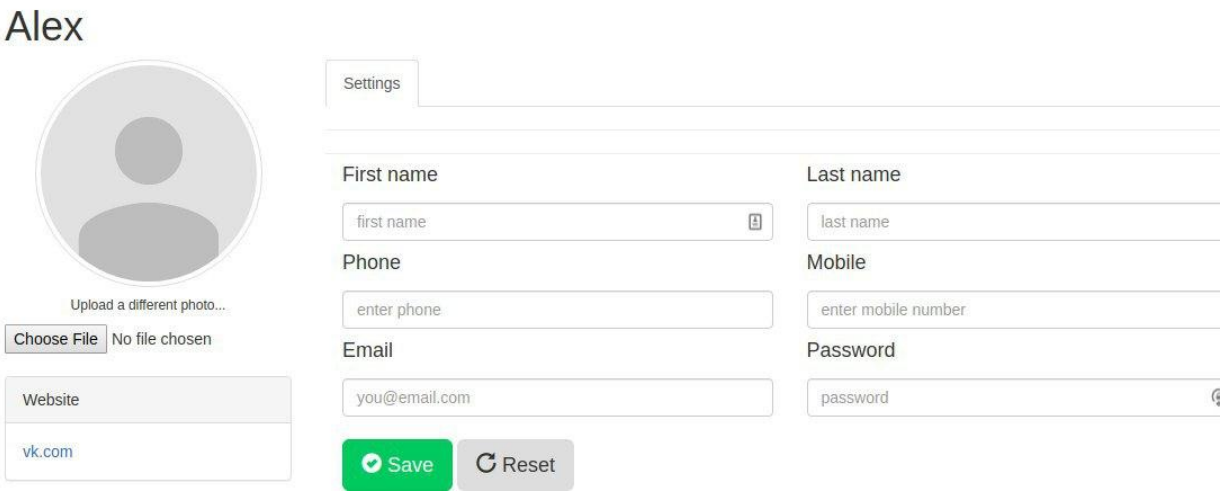


Рис. 4.10. Форма редагування особистих даних

4.2 Інструкція для адміністратора

Для адміністратора існує окремий вхід у так звану адміністративну частину сайту

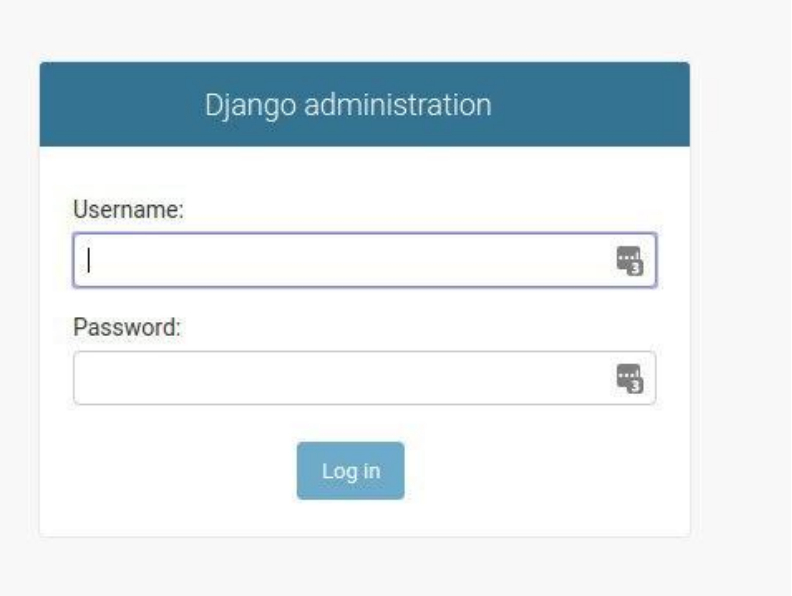


Рис. 4.11. Вхід для адміністратора

Після входу у адміністративну частину адміністратору доступний дашборд із багатьма функціями.

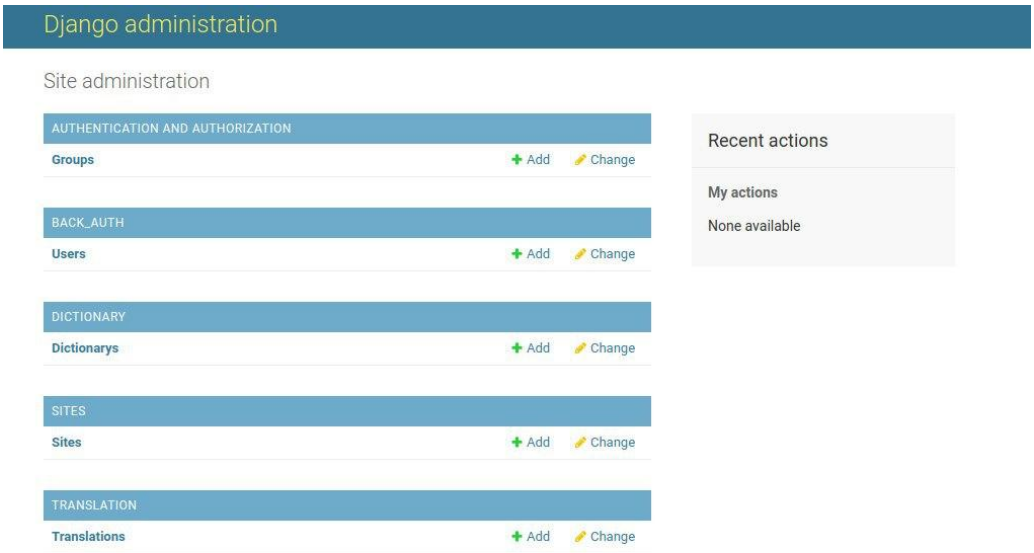


Рис. 4.12. Дашборд адміністратора

Адміністратор може передивлятися список користувачів у системі відкривши відповідний розділ в адмін-панелі.

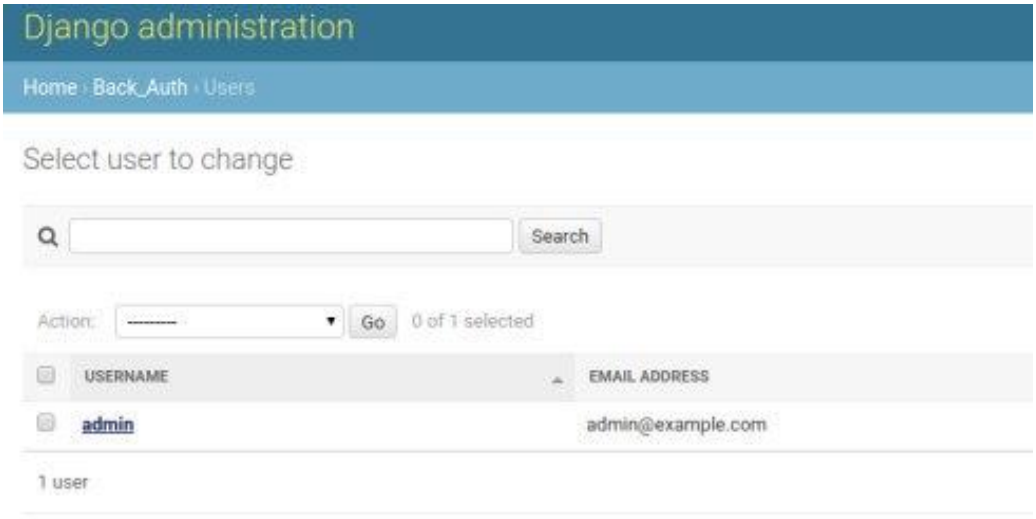


Рис. 4.13. Список користувачів

Окрім можливості передивитися список користувачів адміністратор має можливість видалення будь-якого з них

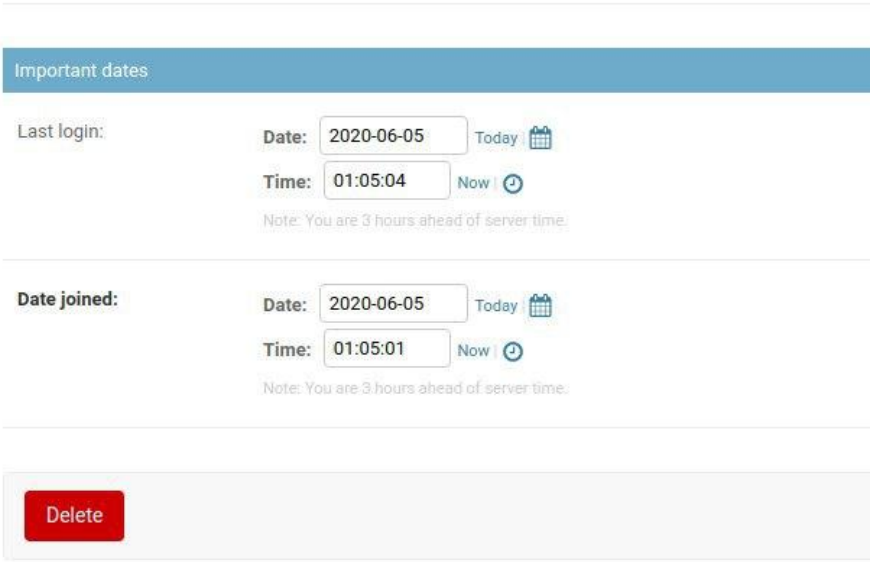


Рис. 4.14. Видалення користувача

Видаленням користувачів права адміністратора не завершуються, він також можна редагувати будь-якого з користувачів. Додавати їх у групи, додавати їм права та інше.

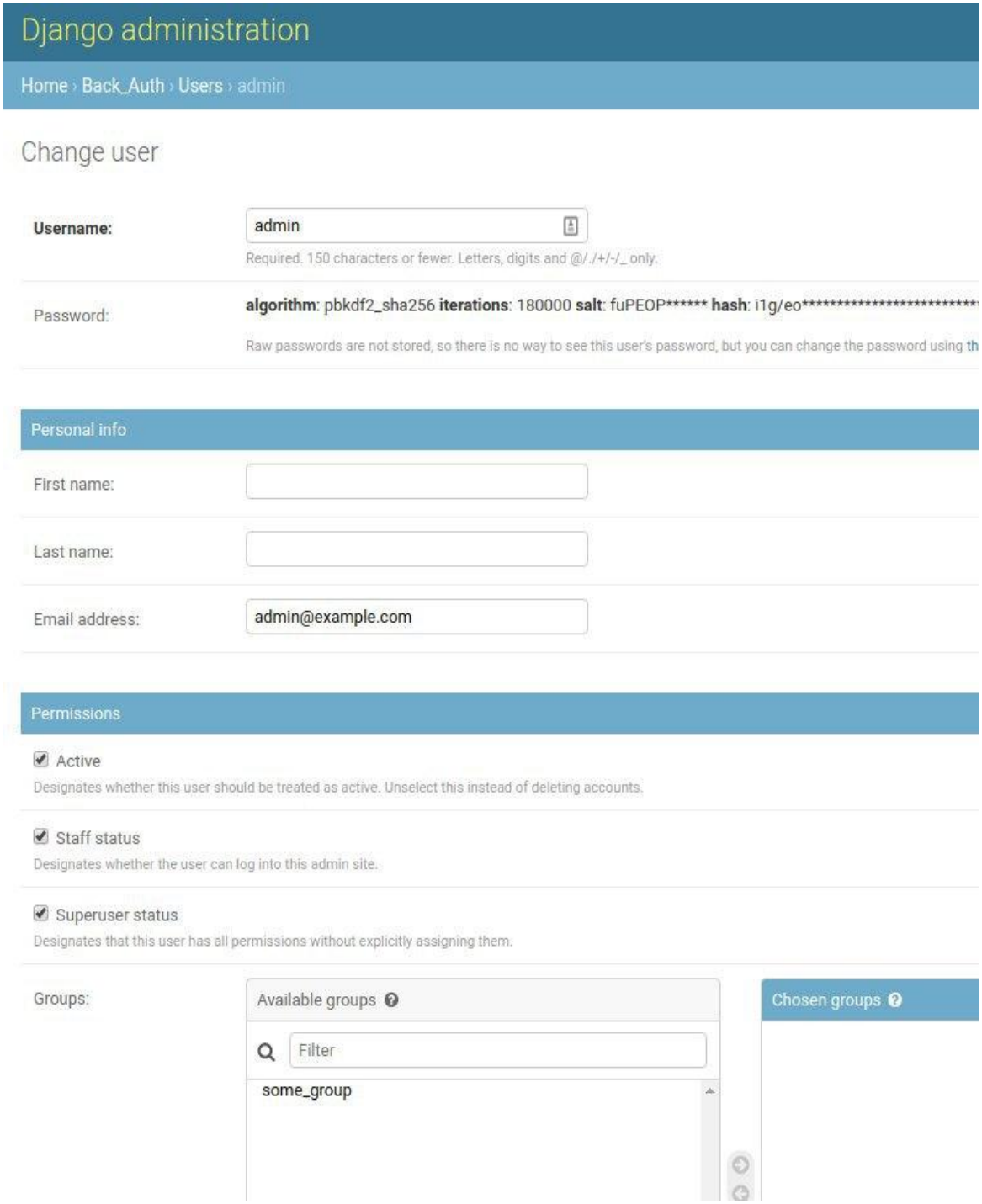


Рис. 4.15. Редагування користувача

Адміністратор може додати користувача у систему натиснувши на відповідний елемент меню

Рис. 4.16. Додавання користувачів

Як було сказано вище, користувачі можуть належати до певної групи, групами також керує адміністратор, він може їх створювати, редагувати та видаляти.

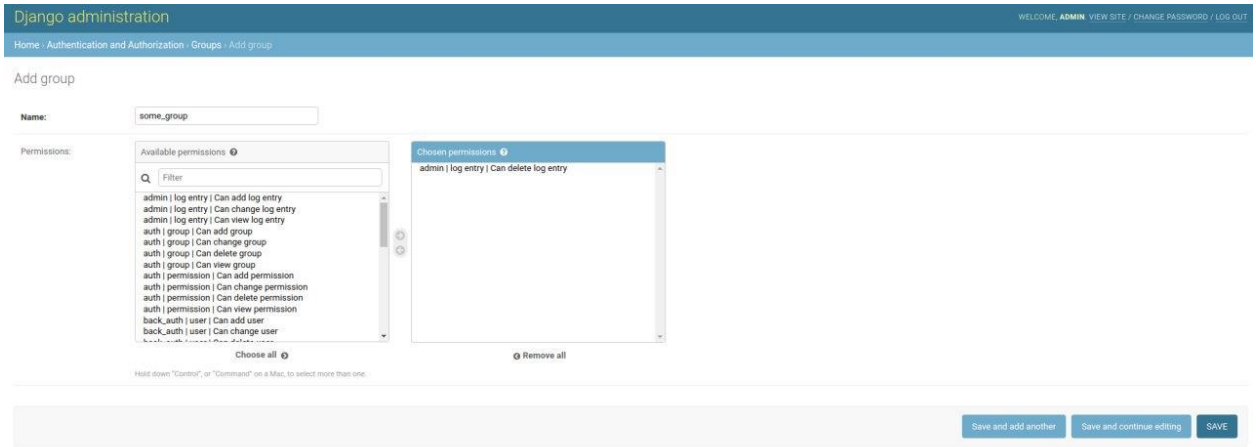


Рис. 4.17. Додавання групи

Рис. 4.18. Редагування групи

Також є можливість додавання перекладу із панелі адміністратора, щоб це зробити необхідно натиснути на відповідний пункт меню

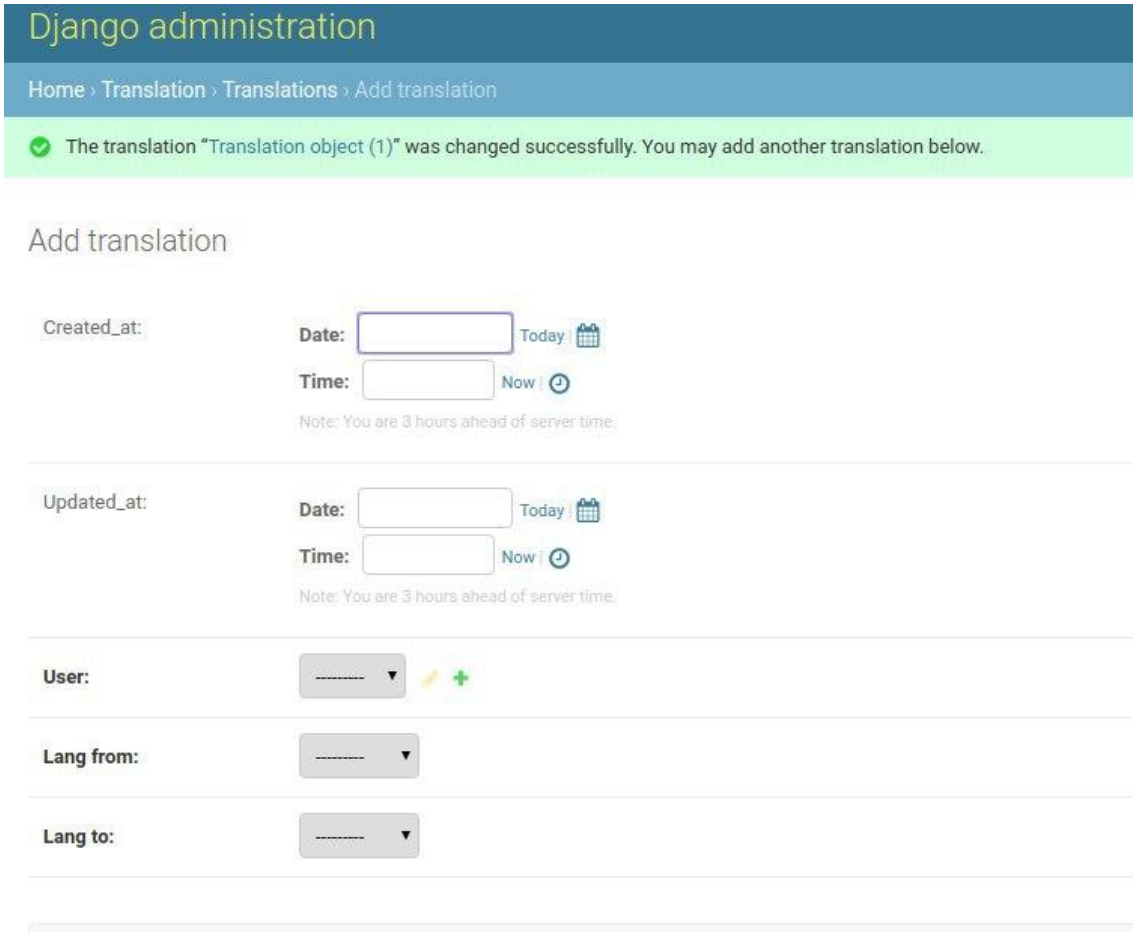


Рис. 4.19. Створення перекладу

Якщо перейти у вкладку Translations в панелі адміністратора то там буде відображатися існуючи переклади.

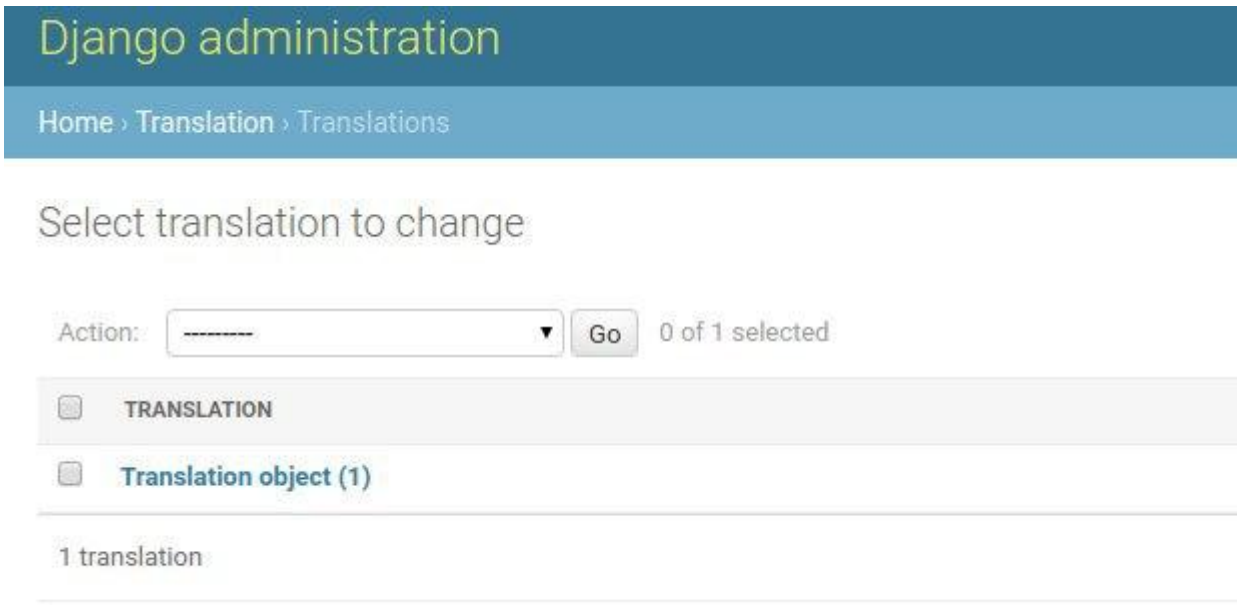


Рис. 4.20. Вибір перекладу для редагування

Адміністратор може редагувати переклади у системі натиснувши на відповідний елемент меню

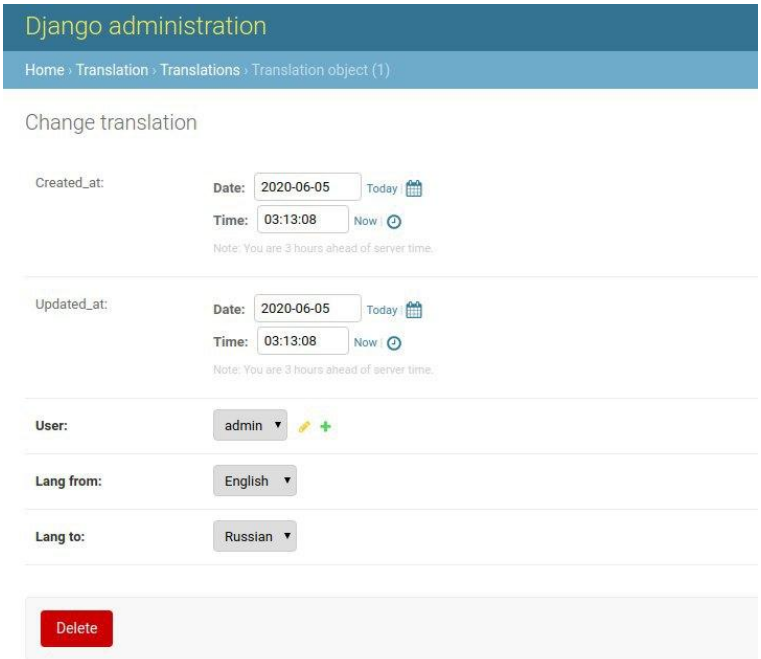


Рис. 4.21. Редагування перекладу

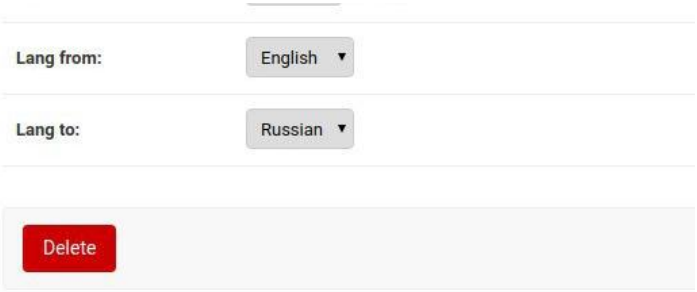


Рис. 4.22. Видалення перекладу

Також є можливість додавання словника із панелі адміністратора, щоб це зробити необхідно натиснути на відповідний пункт меню:

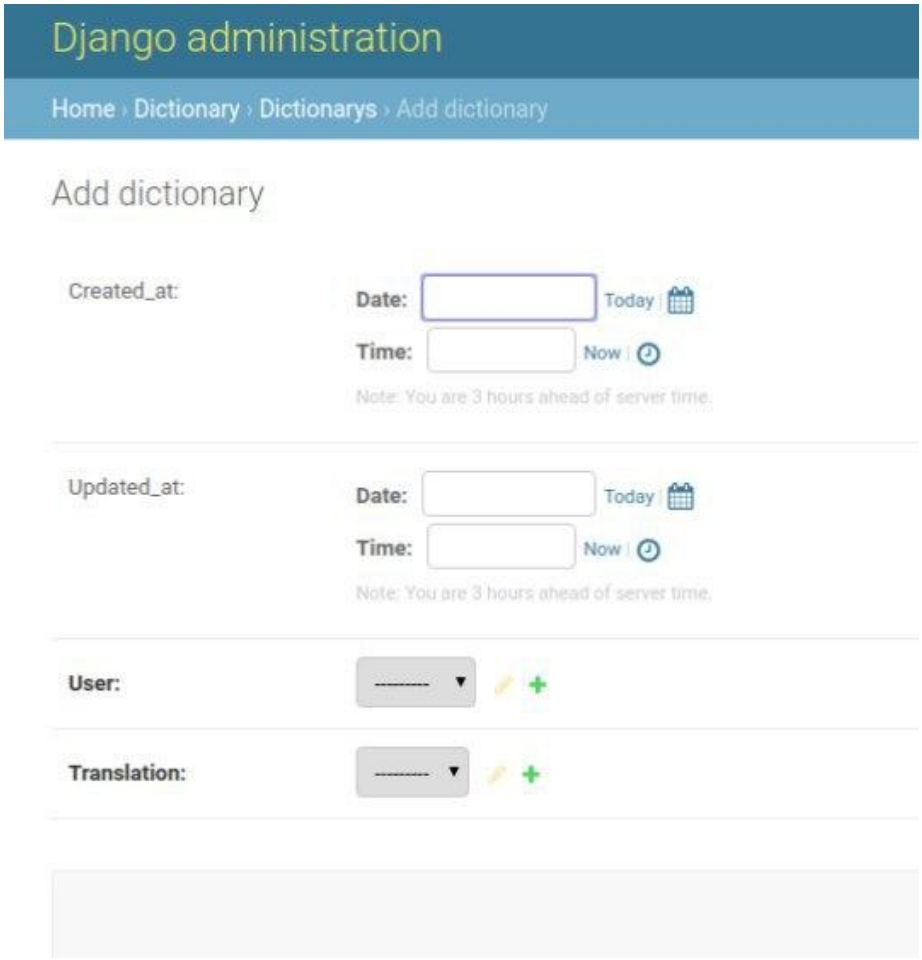


Рис. 4.23. Додавання словника

Якщо перейти у вкладку Dictionaries в панелі адміністратора то там буде відображатися існуючи словники.

Рис. 4.24. Вибір словника для редагування

Адміністратор може редагувати словники у системі натиснувши на відповідний елемент меню

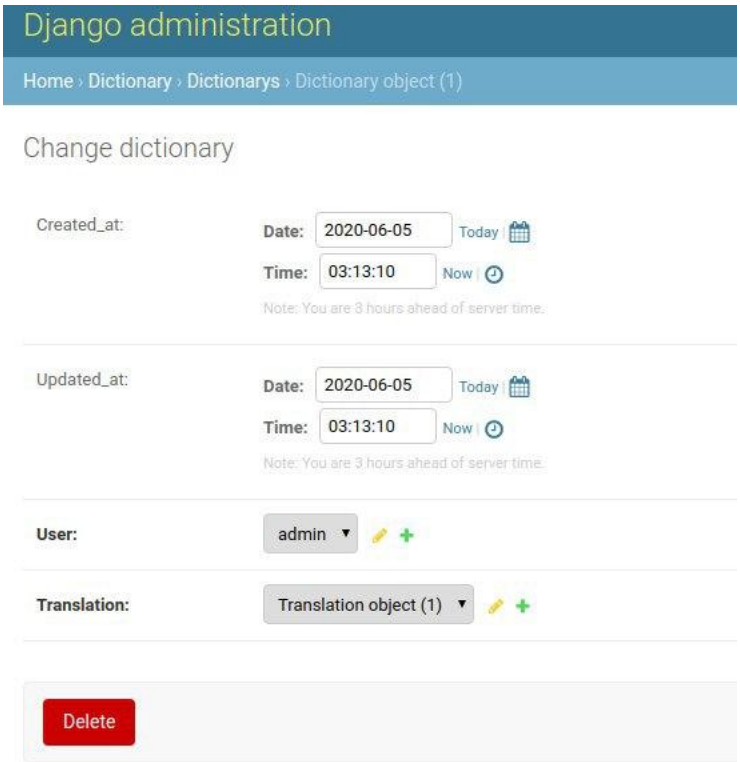


Рис. 4.25. Редагування словника

Також адміністратору доступна дані щодо того як і коли словник був змінений, так яка зміна була здійснена

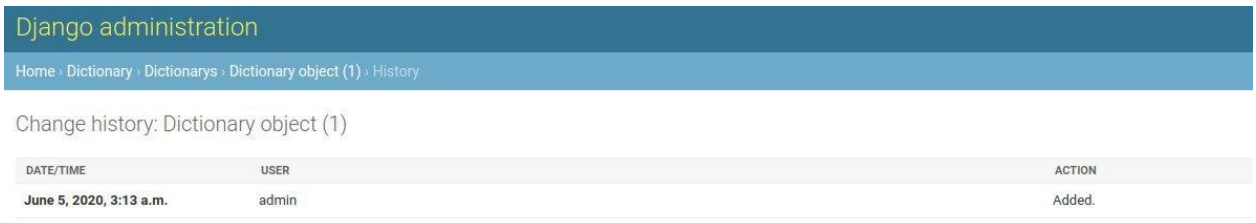


Рис. 4.26. Перегляд історії змін словника

ВИСНОВКИ ДО РОЗДІЛУ 4

Даний розділ було присвячено формуванню інструкції роботи користувача з системою, було описано інтерфейс користувача і описано дії користувача для використання основних функціональних можливостей додатку. Враховуючи поставлені вимоги до системи, було описано всі можливості користувача під час використання програми, які нададуть можливість ефективного користування системою. Для написання системи було використано середу розробки Doom Emacs та операційну систему Linux.

Виходячи з розділу можна зробити висновок, що основними функціональними можливостями додатку є:

- Реєстрація і вхід
- Перегляд словників користувача
- Пошук по словникам у системі
- Створення перекладів із додаванням фото та аудіо
- Створення словників
- Вихід
- Профіль адміністратора із багатьма функціями
- Перегляд свого профілю та інших користувачів
- Відновлення паролю
- Редагування даних профілю
- Видалення перекладів та словників

Висновки

Даний дипломний проект присвячено проектуванню системи дистанційного навчання іноземних мов. Метою проекту було створення веб-додатку для створення перекладів, словників з перекладами, а також багато іншої фунціоналу, що потрібний для ефективного та швидкого вивчення іноземних мов.

Під час виконання проекту було розглянуто основні переваги та недоліки вже існуючих систем, а також було сформульовано вимоги до даного проекту.

Було досліджено предметну область, визначено задачі та вимоги, з якими додаток повинен був справлятись, також було наведено шляхи використання додатку як для користувача так і для адміністратора. Дані були структуровані та зображені у вигляді таблиць, схем та списків. Спроектовано та реалізовано програмний продукт, які повністю готовий до користування.

Виходячи із вимог, які були поставлені проведено аналіз платформ для реалізації додатку та можливостей застосування технологій. Щоб реалізувати усі вимоги було використано багато сучасних технологій для веб-розробки, що було детально описано. У якості мов для розробки було обрано Python для backend частини системи та JavaScript для frontend частини. Також були проаналізовані допоміжні бібліотеки для написання веб-додатку, обґрунтовано та визначено доцільність їх використання.

В рамках додатку реалізовано заздалегідь визначений функціонал.

В рамках виконання роботи проведено:

- Огляд існуючих рішень таких систем
- Проектування усіх частин системи, що в подальшому було реалізовано, включаючи: проектування бази даних, архітектуру frontend та backend, вибір сервісів для розгортки, а також фреймворк для тестування.
- Розробку модуля регістрації та авторизації користувача.

- Розробку модуля налаштувань для вибіру мови, якою користувач хоче оволодіти.
- Розробку інтеграції із зовнішнім перекладачем.
- Можливість створення словників слів з можливістю додавання, редагування та видалення перекладів.
- Алгоритм, завдяки якому система визначає чи вивчив користувач іноземне слово чи ні.

- Інтеграція з AWS S3 для збереження медіа-файлів
- Наведено інструкцію щодо роботи користувача з програмою

В результаті виконання роботи виконано ці заходи та розроблено програмне забезпечення системи дистанційного навчання іноземної мови з веб-інтерфейсом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Середа В. В. Top-besplatnykh-sistem-distancionnogo-obucheniya-personala [Електронний ресурс] / Владислав Васильович Середа. – 2018. – Режим доступу до ресурсу: http://hr-elearning.ru/top-besplatnykh-sistem-distancionnogo-obucheniya-personala/.
2. Ewaryst T. Internet - Technical Development and Applications / T. Ewaryst, K. Adrian. – Berlin: Springer Science & Business Media, 2009. – 284 с.
3. SOLID [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/348286/.
4. REST API Best Practices [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/351890/.
5. Reza H. The Digital University - Building a Learning Community / H. Reza, H. Stephen. – Berlin: Springer Science & Business Media, 2008. – 252 с.
6. Why We Use Django Framework & What Is Django Used For [Електронний ресурс] // Django Stars EDITORIAL TEAM – Режим доступу до ресурсу: https://djangostars.com/blog/why-we-use-django-framework/.
7. Youssef N. What is Django? Advantages and Disadvantages [Електронний ресурс] / Nader Youssef. – 2020. – Режим доступу до ресурсу: https://hackr.io/blog/what-is-django-advantages-and-disadvantages-of-using-django.
8. Д. В. Іртегов. Введення в операційні системи. / Д. В. Іртегов. - 2-е вид. - СПб. : БХВ-Петербург, 2012. - 1040
9. FLASK PYTHON FRAMEWORK [Електронний ресурс] // Quintagroup – Режим доступу до ресурсу: https://quintagroup.com/cms/python/flask.

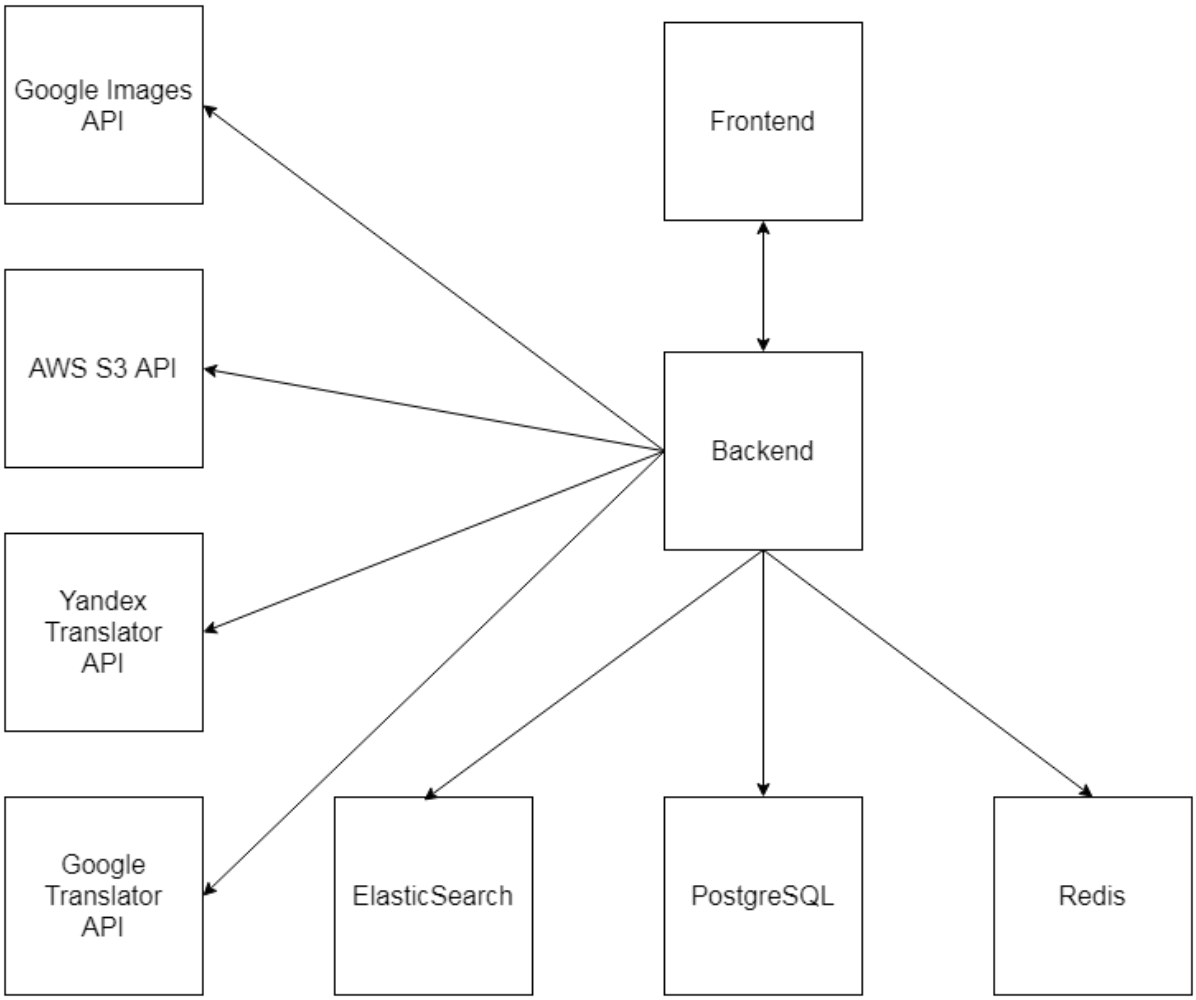
10. Web and Database Security [Електронний ресурс] / X.Jiping, X. Lifeng, Z. Jian, H. Tao. – 2012. – Режим доступу до ресурсу: https://www.intechopen.com/books/security-enhanced-applications-for-information-systems/web-and-database-security.
11. Webpack 4 — The Complete Guide [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://medium.com/better-programming/webpack-4- the-complete-guide-af1b1e2e3f7a.
12. Angular vs React vs Vue: Which Framework to Choose in 2020 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.
13. React или Angular или Vue.js — что выбрать? [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/post/476312/.
14. API Development for Everyone [Електронний ресурс] – Режим доступу до ресурсу: https://swagger.io/
15. Таненбаум. Сучасні операційні системи / Е. Таненбаум. - 2-е вид. - СПб. та ін.: ПІТЕР, 2002. - 1040 с
16. Django Documentation [Електронний ресурс] // Django Software Foundation. – 2020. – Режим доступу до ресурсу: https://buildmedia.readthedocs.org/media/pdf/django/3.0.x/django.pdf.

СТРУКТУРНА СХЕМА

до дипломного проекту

На тему «Система дистанційного навчання іноземної мови»

СТРУКТУРНА СХЕМА



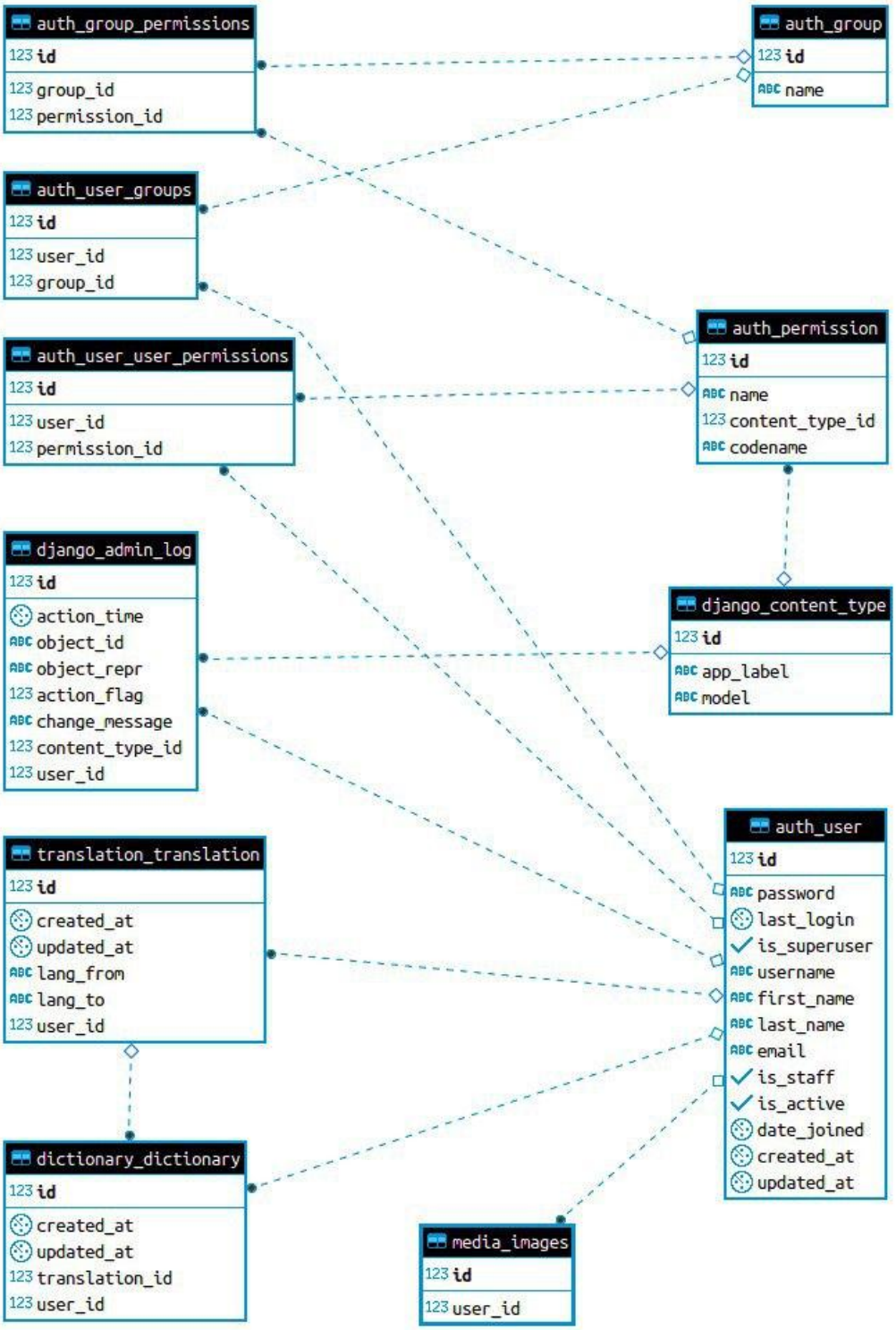
					ДП.6505.04.000 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Горбов О.Ю.			*Система дистанційного навчання іноземної мови* *Додаток 1*	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Виноградов Ю.М.					
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-62	
Н. Контр.		Сімоненко В. П.					
Затвердив							

ФУНКЦІОНАЛЬНА СХЕМА

до дипломного проекту

На тему «Система дистанційного навчання іноземної мови»

ФУНКЦІОНАЛЬНА СХЕМА



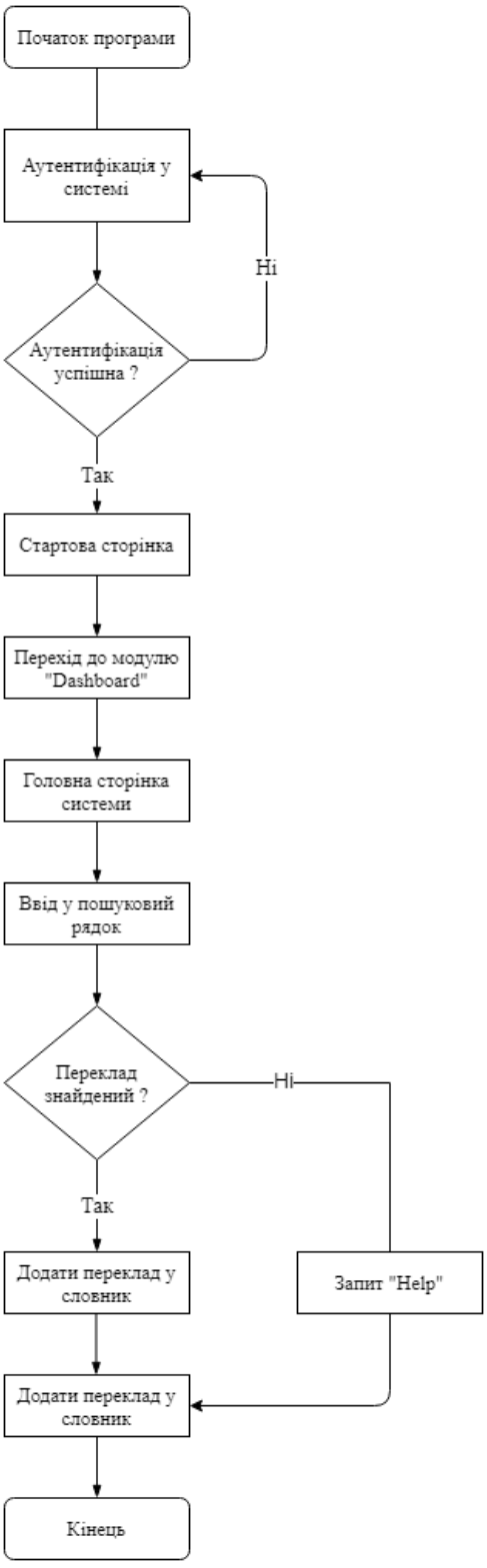
					ДП.6505.05.000 Д2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Горбов О.Ю.			*Система дистанційного навчання іноземної мови* *Додаток 2*	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Виноградов Ю.М					
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-62	
Н. Контр.		Сімоненко В. П.					
Затвердив							

ПРИНЦИПОВА СХЕМА

до дипломного проєкту

На тему «Система дистанційного навчання іноземної мови»

ПРИНЦИПОВА СХЕМА



					ДП.6505.06.000 Д3		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Горбов О.Ю.			*Система дистанційного навчання іноземної мови* *Додаток 3*	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Виноградов Ю.М.					
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-62	
Н. Контр.		Сімоненко В. П.					
Затвердив							